

**Umbra**  
a-01

Generated by Doxygen 1.5.5

Tue Jun 10 13:51:20 2008



# Contents

<b>1 Namespace Index</b>	<b>1</b>
1.1 Package List . . . . .	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class Hierarchy . . . . .	3
<b>3 Class Index</b>	<b>7</b>
3.1 Class List . . . . .	7
<b>4 File Index</b>	<b>11</b>
4.1 File List . . . . .	11
<b>5 Namespace Documentation</b>	<b>15</b>
5.1 Package umbra . . . . .	15
5.2 Package umbra.editor . . . . .	16
5.3 Package umbra.editor.actions . . . . .	17
5.4 Package umbra.editor.actions.history . . . . .	18
5.5 Package umbra.editor.actions.info . . . . .	19
5.6 Package umbra.editor.parsing . . . . .	20
5.7 Package umbra.instructions . . . . .	21
5.8 Package umbra.instructions.ast . . . . .	23
5.9 Package umbra.java . . . . .	25
5.10 Package umbra.java.actions . . . . .	26
5.11 Package umbra.lib . . . . .	27
<b>6 Class Documentation</b>	<b>29</b>
6.1 umbra.instructions.ast.AnnotationLineController Class Reference . . . . .	29
6.2 umbra.instructions.ast.ArithmeticInstruction Class Reference . . . . .	33
6.3 umbra.instructions.ast.ArrayInstruction Class Reference . . . . .	42
6.4 umbra.lib.BMLParsing Class Reference . . . . .	47

6.5	umbra.editor.parsing.BytecodeBMLSecScanner Class Reference . . . . .	50
6.6	umbra.editor.actions.BytecodeColorAction Class Reference . . . . .	53
6.7	umbra.editor.actions.BytecodeCombineAction Class Reference . . . . .	56
6.8	umbra.instructions.BytecodeCommentParser Class Reference . . . . .	63
6.9	umbra.editor.BytecodeConfiguration Class Reference . . . . .	74
6.10	umbra.editor.BytecodeContribution Class Reference . . . . .	80
6.11	umbra.editor.BytecodeContribution.BytecodeListener Class Reference . . . . .	85
6.12	umbra.instructions.BytecodeController Class Reference . . . . .	90
6.13	umbra.instructions.BytecodeControllerComments Class Reference . . . . .	93
6.14	umbra.instructions.BytecodeControllerContainer Class Reference . . . . .	96
6.15	umbra.instructions.BytecodeControllerHelper Class Reference . . . . .	105
6.16	umbra.editor.BytecodeDocument Class Reference . . . . .	112
6.17	umbra.editor.BytecodeDocumentProvider Class Reference . . . . .	121
6.18	umbra.editor.actions.BytecodeDoubleClickStrategy Class Reference . . . . .	124
6.19	umbra.editor.BytecodeEditor Class Reference . . . . .	127
6.20	umbra.editor.actions.BytecodeEditorAction Class Reference . . . . .	136
6.21	umbra.editor.BytecodeEditorContributor Class Reference . . . . .	141
6.22	umbra.instructions.ast.BytecodeLineController Class Reference . . . . .	151
6.23	umbra.editor.parsing.BytecodePartitionScanner Class Reference . . . . .	160
6.24	umbra.editor.actions.BytecodeRebuildAction Class Reference . . . . .	164
6.25	umbra.editor.actions.BytecodeRefreshAction Class Reference . . . . .	167
6.26	umbra.editor.actions.history.BytecodeRestoreAction Class Reference . . . . .	171
6.27	umbra.editor.parsing.BytecodeScanner Class Reference . . . . .	175
6.28	umbra.lib.BytecodeStrings Class Reference . . . . .	183
6.29	umbra.lib.BytecodeStringsGeneric Class Reference . . . . .	187
6.30	umbra.lib.BytecodeStringsMnemonics Class Reference . . . . .	190
6.31	umbra.editor.actions.BytecodeSynchrAction Class Reference . . . . .	197
6.32	umbra.instructions.BytecodeTextParser Class Reference . . . . .	201
6.33	umbra.editor.parsing.BytecodeWhitespaceDetector Class Reference . . . . .	210
6.34	umbra.editor.parsing.BytecodeWordDetector Class Reference . . . . .	212
6.35	umbra.instructions.CannotCallRuleException Class Reference . . . . .	214
6.36	umbra.editor.actions.history.ClearHistoryAction Class Reference . . . . .	215
6.37	umbra.editor.parsing.ColorManager Class Reference . . . . .	217
6.38	umbra.editor.ColorModeContainer Class Reference . . . . .	220
6.39	umbra.editor.parsing.ColorValues Class Reference . . . . .	223
6.40	umbra.instructions.ast.CommentLineController Class Reference . . . . .	228

6.41 umbra.java.actions.CommitAction Class Reference . . . . .	231
6.42 umbra.instructions.ast.ConversionInstruction Class Reference . . . . .	234
6.43 umbra.java.actions.DisasBCEL Class Reference . . . . .	240
6.44 umbra.instructions.DispatchingAutomaton Class Reference . . . . .	245
6.45 umbra.editor.DocumentSynchroniser Class Reference . . . . .	247
6.46 umbra.lib.EclipseIdentifiers Class Reference . . . . .	251
6.47 umbra.instructions.ast.EmptyLineController Class Reference . . . . .	253
6.48 umbra.instructions.ast.FieldInstruction Class Reference . . . . .	255
6.49 umbra.lib.FileNames Class Reference . . . . .	259
6.50 umbra.instructions.FragmentParser Class Reference . . . . .	269
6.51 umbra.lib.GUIMessages Class Reference . . . . .	271
6.52 umbra.instructions.ast.HeaderLineController Class Reference . . . . .	279
6.53 umbra.editor.actions.history.HistoryAction Class Reference . . . . .	282
6.54 umbra.lib.HistoryOperations Class Reference . . . . .	284
6.55 umbra.instructions.ast.IConstInstruction Class Reference . . . . .	291
6.56 umbra.instructions.ast.IincInstruction Class Reference . . . . .	295
6.57 umbra.instructions.InitParser Class Reference . . . . .	300
6.58 umbra.editor.actions.info.InstallInfoAction Class Reference . . . . .	306
6.59 umbra.instructions.ast.InstructionLineController Class Reference . . . . .	308
6.60 umbra.instructions.InstructionNameParser Class Reference . . . . .	318
6.61 umbra.instructions.InstructionParser Class Reference . . . . .	324
6.62 umbra.instructions.InstructionParserGeneric Class Reference . . . . .	328
6.63 umbra.instructions.InstructionParserHelper Class Reference . . . . .	333
6.64 umbra.instructions.InstructionTypeParser Class Reference . . . . .	340
6.65 umbra.instructions.ast.InvokeInstruction Class Reference . . . . .	345
6.66 umbra.instructions.ast.JumpInstruction Class Reference . . . . .	349
6.67 umbra.instructions.ast.LdcInstruction Class Reference . . . . .	359
6.68 umbra.instructions.LineContext Class Reference . . . . .	363
6.69 umbra.instructions.ast.LoadStoreArrayInstruction Class Reference . . . . .	372
6.70 umbra.instructions.ast.LoadStoreConstInstruction Class Reference . . . . .	381
6.71 umbra.instructions.ast.MultiInstruction Class Reference . . . . .	389
6.72 umbra.instructions.ast.NewInstruction Class Reference . . . . .	393
6.73 umbra.editor.parsing.NonRuleBasedDamagerRepairer Class Reference . . . . .	397
6.74 umbra.instructions.ast.NumInstruction Class Reference . . . . .	402
6.75 umbra.instructions.ast.OtherInstruction Class Reference . . . . .	406
6.76 umbra.instructions.Parsing Class Reference . . . . .	408

6.77	umbra.instructions.ast.PushInstruction Class Reference . . . . .	413
6.78	umbra.instructions.ast.SingleInstruction Class Reference . . . . .	417
6.79	umbra.editor.parsing.SpecialNumberRule Class Reference . . . . .	425
6.80	umbra.instructions.ast.StackInstruction Class Reference . . . . .	428
6.81	umbra.instructions.ast.StringInstruction Class Reference . . . . .	436
6.82	umbra.java.actions.SynchrSBAAction Class Reference . . . . .	438
6.83	umbra.instructions.ast.ThrowsLineController Class Reference . . . . .	442
6.84	umbra.editor.parsing.TokenGetter Class Reference . . . . .	444
6.85	umbra.lib.UmbraClassException Class Reference . . . . .	447
6.86	umbra.lib.UmbraException Class Reference . . . . .	449
6.87	umbra.lib.UmbraLocationException Class Reference . . . . .	450
6.88	umbra.lib.UmbraMethodException Class Reference . . . . .	454
6.89	umbra.UmbraPlugin Class Reference . . . . .	456
6.90	umbra.lib.UmbraRangeException Class Reference . . . . .	459
6.91	umbra.lib.UmbraRuntimeException Class Reference . . . . .	461
6.92	umbra.lib.UmbraSynchronisationException Class Reference . . . . .	462
6.93	umbra.instructions.ast.UnclassifiedInstruction Class Reference . . . . .	463
6.94	umbra.instructions.ast.UnknownLineController Class Reference . . . . .	466
6.95	umbra.editor.actions.info.UserGuideAction Class Reference . . . . .	468
<b>7</b>	<b>File Documentation</b> . . . . .	<b>471</b>
7.1	source/umbra/editor/actions/BytecodeColorAction.java File Reference . . . . .	471
7.2	source/umbra/editor/actions/BytecodeCombineAction.java File Reference . . . . .	472
7.3	source/umbra/editor/actions/BytecodeDoubleClickStrategy.java File Reference . . . . .	473
7.4	source/umbra/editor/actions/BytecodeEditorAction.java File Reference . . . . .	474
7.5	source/umbra/editor/actions/BytecodeRebuildAction.java File Reference . . . . .	475
7.6	source/umbra/editor/actions/BytecodeRefreshAction.java File Reference . . . . .	476
7.7	source/umbra/editor/actions/BytecodeSynchrAction.java File Reference . . . . .	477
7.8	source/umbra/editor/actions/history/BytecodeRestoreAction.java File Reference . . . . .	478
7.9	source/umbra/editor/actions/history/ClearHistoryAction.java File Reference . . . . .	479
7.10	source/umbra/editor/actions/history/HistoryAction.java File Reference . . . . .	480
7.11	source/umbra/editor/actions/info/InstalInfoAction.java File Reference . . . . .	481
7.12	source/umbra/editor/actions/info/UserGuideAction.java File Reference . . . . .	482
7.13	source/umbra/editor/BytecodeConfiguration.java File Reference . . . . .	483
7.14	source/umbra/editor/BytecodeContribution.java File Reference . . . . .	484
7.15	source/umbra/editor/BytecodeDocument.java File Reference . . . . .	485
7.16	source/umbra/editor/BytecodeDocumentProvider.java File Reference . . . . .	486

7.17	source/umbra/editor/BytecodeEditor.java File Reference . . . . .	487
7.18	source/umbra/editor/BytecodeEditorContributor.java File Reference . . . . .	488
7.19	source/umbra/editor/ColorModeContainer.java File Reference . . . . .	489
7.20	source/umbra/editor/DocumentSynchroniser.java File Reference . . . . .	490
7.21	source/umbra/editor/parsing/BytecodeBMLSecScanner.java File Reference . . . . .	491
7.22	source/umbra/editor/parsing/BytecodePartitionScanner.java File Reference . . . . .	492
7.23	source/umbra/editor/parsing/BytecodeScanner.java File Reference . . . . .	493
7.24	source/umbra/editor/parsing/BytecodeWhitespaceDetector.java File Reference . . . . .	494
7.25	source/umbra/editor/parsing/BytecodeWordDetector.java File Reference . . . . .	495
7.26	source/umbra/editor/parsing/ColorManager.java File Reference . . . . .	496
7.27	source/umbra/editor/parsing/ColorValues.java File Reference . . . . .	497
7.28	source/umbra/editor/parsing/NonRuleBasedDamagerRepairer.java File Reference . . . . .	498
7.29	source/umbra/editor/parsing/SpecialNumberRule.java File Reference . . . . .	499
7.30	source/umbra/editor/parsing TokenNameGetter.java File Reference . . . . .	500
7.31	source/umbra/instructions/ast/AnnotationLineController.java File Reference . . . . .	501
7.32	source/umbra/instructions/ast/ArithmeticInstruction.java File Reference . . . . .	502
7.33	source/umbra/instructions/ast/ArrayInstruction.java File Reference . . . . .	503
7.34	source/umbra/instructions/ast/BytecodeLineController.java File Reference . . . . .	504
7.35	source/umbra/instructions/ast/CommentLineController.java File Reference . . . . .	505
7.36	source/umbra/instructions/ast/ConversionInstruction.java File Reference . . . . .	506
7.37	source/umbra/instructions/ast/EmptyLineController.java File Reference . . . . .	507
7.38	source/umbra/instructions/ast/FieldInstruction.java File Reference . . . . .	508
7.39	source/umbra/instructions/ast/HeaderLineController.java File Reference . . . . .	509
7.40	source/umbra/instructions/ast/IConstInstruction.java File Reference . . . . .	510
7.41	source/umbra/instructions/ast/IincInstruction.java File Reference . . . . .	511
7.42	source/umbra/instructions/ast/InstructionLineController.java File Reference . . . . .	512
7.43	source/umbra/instructions/ast/InvokeInstruction.java File Reference . . . . .	513
7.44	source/umbra/instructions/ast/JumpInstruction.java File Reference . . . . .	514
7.45	source/umbra/instructions/ast/LdcInstruction.java File Reference . . . . .	515
7.46	source/umbra/instructions/ast/LoadStoreArrayInstruction.java File Reference . . . . .	516
7.47	source/umbra/instructions/ast/LoadStoreConstInstruction.java File Reference . . . . .	517
7.48	source/umbra/instructions/ast/MultiInstruction.java File Reference . . . . .	518
7.49	source/umbra/instructions/ast/NewInstruction.java File Reference . . . . .	519
7.50	source/umbra/instructions/ast/NumInstruction.java File Reference . . . . .	520
7.51	source/umbra/instructions/ast/OtherInstruction.java File Reference . . . . .	521
7.52	source/umbra/instructions/ast/PushInstruction.java File Reference . . . . .	522

7.53 source/umbra/instructions/ast/SingleInstruction.java File Reference . . . . .	523
7.54 source/umbra/instructions/ast/StackInstruction.java File Reference . . . . .	524
7.55 source/umbra/instructions/ast/StringInstruction.java File Reference . . . . .	525
7.56 source/umbra/instructions/ast/ThrowsLineController.java File Reference . . . . .	526
7.57 source/umbra/instructions/ast/UnclassifiedInstruction.java File Reference . . . . .	527
7.58 source/umbra/instructions/ast/UnknownLineController.java File Reference . . . . .	528
7.59 source/umbra/instructions/BytecodeCommentParser.java File Reference . . . . .	529
7.60 source/umbra/instructions/BytecodeController.java File Reference . . . . .	530
7.61 source/umbra/instructions/BytecodeControllerComments.java File Reference . . . . .	531
7.62 source/umbra/instructions/BytecodeControllerContainer.java File Reference . . . . .	532
7.63 source/umbra/instructions/BytecodeControllerHelper.java File Reference . . . . .	533
7.64 source/umbra/instructions/BytecodeTextParser.java File Reference . . . . .	534
7.65 source/umbra/instructions/CannotCallRuleException.java File Reference . . . . .	535
7.66 source/umbra/instructions/DispatchingAutomaton.java File Reference . . . . .	536
7.67 source/umbra/instructions/FragmentParser.java File Reference . . . . .	537
7.68 source/umbra/instructions/InitParser.java File Reference . . . . .	538
7.69 source/umbra/instructions/InstructionNameParser.java File Reference . . . . .	539
7.70 source/umbra/instructions/InstructionParser.java File Reference . . . . .	540
7.71 source/umbra/instructions/InstructionParserGeneric.java File Reference . . . . .	541
7.72 source/umbra/instructions/InstructionParserHelper.java File Reference . . . . .	542
7.73 source/umbra/instructions/InstructionTypeParser.java File Reference . . . . .	543
7.74 source/umbra/instructions/LineContext.java File Reference . . . . .	544
7.75 source/umbra/instructions/Preparsing.java File Reference . . . . .	545
7.76 source/umbra/java/actions/CommitAction.java File Reference . . . . .	546
7.77 source/umbra/java/actions/DisasBCEL.java File Reference . . . . .	547
7.78 source/umbra/java/actions/SynchrSBAction.java File Reference . . . . .	548
7.79 source/umbra/lib/BMLParsing.java File Reference . . . . .	549
7.80 source/umbra/lib/BytecodeStrings.java File Reference . . . . .	550
7.81 source/umbra/lib/BytecodeStringsGeneric.java File Reference . . . . .	551
7.82 source/umbra/lib/BytecodeStringsMnemonics.java File Reference . . . . .	552
7.83 source/umbra/lib/EclipseIdentifiers.java File Reference . . . . .	553
7.84 source/umbra/lib/FileNames.java File Reference . . . . .	554
7.85 source/umbra/lib/GUIMessages.java File Reference . . . . .	555
7.86 source/umbra/lib/HistoryOperations.java File Reference . . . . .	556
7.87 source/umbra/lib/UmbraClassException.java File Reference . . . . .	557
7.88 source/umbra/lib/UmbraException.java File Reference . . . . .	558

7.89 source/umbra/lib/UmbraLocationException.java File Reference . . . . .	559
7.90 source/umbra/lib/UmbraMethodException.java File Reference . . . . .	560
7.91 source/umbra/lib/UmbraRangeException.java File Reference . . . . .	561
7.92 source/umbra/lib/UmbraRuntimeException.java File Reference . . . . .	562
7.93 source/umbra/lib/UmbraSynchronisationException.java File Reference . . . . .	563
7.94 source/umbra/UmbraPlugin.java File Reference . . . . .	564



# Chapter 1

## Namespace Index

### 1.1 Package List

Here are the packages with brief descriptions (if available):

<a href="#">umbra</a>	15
<a href="#">umbra.editor</a>	16
<a href="#">umbra.editor.actions</a>	17
<a href="#">umbra.editor.actions.history</a>	18
<a href="#">umbra.editor.actions.info</a>	19
<a href="#">umbra.editor.parsing</a>	20
<a href="#">umbra.instructions</a>	21
<a href="#">umbra.instructions.ast</a>	23
<a href="#">umbra.java</a>	25
<a href="#">umbra.java.actions</a>	26
<a href="#">umbra.lib</a>	27



# Chapter 2

## Class Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

umbra.lib.BMLParsing . . . . .	47
umbra.editor.parsing.BytecodeBMLSecScanner . . . . .	50
umbra.editor.BytecodeConfiguration . . . . .	74
umbra.editor.BytecodeContribution . . . . .	80
umbra.editor.BytecodeContribution.BytecodeListener . . . . .	85
umbra.instructions.BytecodeControllerHelper . . . . .	105
umbra.instructions.BytecodeControllerComments . . . . .	93
umbra.instructions.BytecodeControllerContainer . . . . .	96
umbra.instructions.BytecodeController . . . . .	90
umbra.editor.BytecodeDocument . . . . .	112
umbra.editor.BytecodeDocumentProvider . . . . .	121
umbra.editor.actions.BytecodeDoubleClickStrategy . . . . .	124
umbra.editor.BytecodeEditor . . . . .	127
umbra.editor.actions.BytecodeEditorAction . . . . .	136
umbra.editor.actions.BytecodeColorAction . . . . .	53
umbra.editor.actions.BytecodeCombineAction . . . . .	56
umbra.editor.actions.BytecodeRebuildAction . . . . .	164
umbra.editor.actions.BytecodeRefreshAction . . . . .	167
umbra.editor.actions.BytecodeSynchrAction . . . . .	197
umbra.editor.actions.history.BytecodeRestoreAction . . . . .	171
umbra.editor.actions.history.ClearHistoryAction . . . . .	215
umbra.editor.actions.history.HistoryAction . . . . .	282
umbra.editor.BytecodeEditorContributor . . . . .	141
umbra.instructions.ast.BytecodeLineController . . . . .	151
umbra.instructions.ast.CommentLineController . . . . .	228
umbra.instructions.ast.AnnotationLineController . . . . .	29
umbra.instructions.ast.EmptyLineController . . . . .	253
umbra.instructions.ast.HeaderLineController . . . . .	279
umbra.instructions.ast.InstructionLineController . . . . .	308
umbra.instructions.ast.MultiInstruction . . . . .	389
umbra.instructions.ast.NumInstruction . . . . .	402
umbra.instructions.ast.IincInstruction . . . . .	295

umbra.instructions.ast.JumpInstruction . . . . .	349
umbra.instructions.ast.PushInstruction . . . . .	413
umbra.instructions.ast.StackInstruction . . . . .	428
umbra.instructions.ast.OtherInstruction . . . . .	406
umbra.instructions.ast.LdcInstruction . . . . .	359
umbra.instructions.ast.StringInstruction . . . . .	436
umbra.instructions.ast.ArrayInstruction . . . . .	42
umbra.instructions.ast.FieldInstruction . . . . .	255
umbra.instructions.ast.InvokeInstruction . . . . .	345
umbra.instructions.ast.NewInstruction . . . . .	393
umbra.instructions.ast.SingleInstruction . . . . .	417
umbra.instructions.ast.ArithmeticInstruction . . . . .	33
umbra.instructions.ast.ConversionInstruction . . . . .	234
umbra.instructions.ast.IConstInstruction . . . . .	291
umbra.instructions.ast.LoadStoreArrayInstruction . . . . .	372
umbra.instructions.ast.LoadStoreConstInstruction . . . . .	381
umbra.instructions.ast.UnclassifiedInstruction . . . . .	463
umbra.instructions.ast.ThrowsLineController . . . . .	442
umbra.instructions.ast.UnknownLineController . . . . .	466
umbra.editor.parsing.BytecodePartitionScanner . . . . .	160
umbra.editor.parsing.BytecodeScanner . . . . .	175
umbra.lib.BytecodeStringsGeneric . . . . .	187
umbra.lib.BytecodeStringsMnemonics . . . . .	190
umbra.lib.BytecodeStrings . . . . .	183
umbra.instructions.BytecodeTextParser . . . . .	201
umbra.instructions.BytecodeCommentParser . . . . .	63
umbra.instructions.FragmentParser . . . . .	269
umbra.instructions.InitParser . . . . .	300
umbra.editor.parsing.BytecodeWhitespaceDetector . . . . .	210
umbra.editor.parsing.BytecodeWordDetector . . . . .	212
umbra.instructions.CannotCallRuleException . . . . .	214
umbra.editor.parsing.ColorManager . . . . .	217
umbra.editor.ColorModeContainer . . . . .	220
umbra.editor.parsing.ColorValues . . . . .	223
umbra.java.actions.CommitAction . . . . .	231
umbra.java.actions.DisasBCEL . . . . .	240
umbra.instructions.DispatchingAutomaton . . . . .	245
umbra.editor.DocumentSynchroniser . . . . .	247
umbra.lib.EclipseIdentifiers . . . . .	251
umbra.lib.FileNames . . . . .	259
umbra.lib.GUIMessages . . . . .	271
umbra.lib.HistoryOperations . . . . .	284
umbra.editor.actions.info.InstalInfoAction . . . . .	306
umbra.instructions.InstructionParserGeneric . . . . .	328
umbra.instructions.InstructionNameParser . . . . .	318
umbra.instructions.InstructionTypeParser . . . . .	340
umbra.instructions.InstructionParser . . . . .	324
umbra.instructions.InstructionParserHelper . . . . .	333
umbra.instructions.LineContext . . . . .	363
umbra.editor.parsing.NonRuleBasedDamagerRepairer . . . . .	397
umbra.instructions.Preparsing . . . . .	408
umbra.editor.parsing.SpecialNumberRule . . . . .	425

umbra.java.actions.SynchrSBAction . . . . .	438
umbra.editor.parsing.TokenGetter . . . . .	444
umbra.lib.UmbraClassException . . . . .	447
umbra.lib.UmbraException . . . . .	449
umbra.lib.UmbraLocationException . . . . .	450
umbra.lib.UmbraMethodException . . . . .	454
umbra.UmbraPlugin . . . . .	456
umbra.lib.UmbraRangeException . . . . .	459
umbra.lib.UmbraRuntimeException . . . . .	461
umbra.lib.UmbraSynchronisationException . . . . .	462
umbra.editor.actions.info.UserGuideAction . . . . .	468



# Chapter 3

## Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<code>umbra.instructions.ast.AnnotationLineController</code> (This class handles the creation and correctness of line controllers that contain BML annotations ) . . . . .	29
<code>umbra.instructions.ast.ArithmeticInstruction</code> (This class handles the creation and correctness for arithmetic <code>instructions</code> with no parameters ) . . . . .	33
<code>umbra.instructions.ast.ArrayInstruction</code> (This class handles the creation and correctness for the instruction to create new arrays of primitive types (newarray) ) . . . . .	42
<code>umbra.lib.BMLParsing</code> (This class is responsible for communication with BMLLib library (except code position synchronization, that calls only stateless, static methods from BML-Lib) ) . . . . .	47
<code>umbra.editor.parsing.BytecodeBMLScanner</code> (This class is responsible for colouring these texts in a byte code <code>editor</code> window which are inside BML annotations areas ) . . . . .	50
<code>umbra.editor.actions.BytecodeColorAction</code> (This class defines an action of changing the coloring style ) . . . . .	53
<code>umbra.editor.actions.BytecodeCombineAction</code> (This is an action associated with a byte code <code>editor</code> (an extension .btc) ) . . . . .	56
<code>umbra.instructions.BytecodeCommentParser</code> (This class handles the operations which are connected with the handling of the comments ) . . . . .	63
<code>umbra.editor.BytecodeConfiguration</code> (This class is used by the <code>BytecodeEditor</code> with the matter of double click strategy and colour versions ) . . . . .	74
<code>umbra.editor.BytecodeContribution</code> (This class represents a GUI element that is contributed to the eclipse GUI by the byte code <code>editor</code> ) . . . . .	80
<code>umbra.editor.BytecodeContribution.BytecodeListener</code> (This is a listener class that receives all the events that change the content of the current byte code document ) . . . . .	85
<code>umbra.instructions.BytecodeController</code> (This class defines some structures related to BCEL as well as to the byte code <code>editor</code> contents ) . . . . .	90
<code>umbra.instructions.BytecodeControllerComments</code> (This class contains the functionality of the <code>BytecodeController</code> class which is responsible for the handling of the end-of-line comments and interline comments ) . . . . .	93
<code>umbra.instructions.BytecodeControllerContainer</code> (This class encapsulates the internal structures of the <code>BytecodeController</code> and gives the internal interface to them ) . . . . .	96
<code>umbra.instructions.BytecodeControllerHelper</code> (This class contains various helper methods that are used in the <code>BytecodeController</code> class ) . . . . .	105
<code>umbra.editor.BytecodeDocument</code> (This class is an abstract model of a byte code textual document )	112

<a href="#">umbra.editor.BytecodeDocumentProvider</a> (This class has been modified with relation to the generated automatically to allow adding listener that is responsible for error checking ) . . . . .	121
<a href="#">umbra.editor.actions.BytecodeDoubleClickStrategy</a> (This class is responsible for action that is performed after a double click event in a byte code <a href="#">editor</a> window ) . . . . .	124
<a href="#">umbra.editor.BytecodeEditor</a> (This is the main class that defines the byte code <a href="#">editor</a> ) . . . . .	127
<a href="#">umbra.editor.actions.BytecodeEditorAction</a> (This class defines the common operations for all the byte code <a href="#">editor</a> actions ) . . . . .	136
<a href="#">umbra.editor.BytecodeEditorContributor</a> (This is managing class that adds <a href="#">actions</a> to workbench menus and toolbars for a byte code <a href="#">editor</a> ) . . . . .	141
<a href="#">umbra.instructions.ast.BytecodeLineController</a> (This is completely abstract class that contains some information useful when the line is modified or BCEL structure is created) . . . . .	151
<a href="#">umbra.editor.parsing.BytecodePartitionScanner</a> (This class is responsible for dividing the byte code document into partitions the colouring of which is governed by different rules) . . . . .	160
<a href="#">umbra.editor.actions.BytecodeRebuildAction</a> (This class defines action of restoring the original version of a class file (it is saved with the name prefixed with '_') and then generating byte code (.btc) directly from it) . . . . .	164
<a href="#">umbra.editor.actions.BytecodeRefreshAction</a> (This is a class defining an action: save current byte code <a href="#">editor</a> window and re-generate byte code from the .class file) . . . . .	167
<a href="#">umbra.editor.actions.history.BytecodeRestoreAction</a> (This class defines action of restoring byte code from <a href="#">history</a> ) . . . . .	171
<a href="#">umbra.editor.parsing.BytecodeScanner</a> (This class is responsible for colouring these texts in a byte code <a href="#">editor</a> window which are outside BML annotations areas) . . . . .	175
<a href="#">umbra.lib.BytecodeStrings</a> (The container for all the byte code and BML strings) . . . . .	183
<a href="#">umbra.lib.BytecodeStringsGeneric</a> (The class is a container for all byte code <a href="#">instructions</a> and the sequences that indicate the starts and ends of comments or BML annotation areas) . . . . .	187
<a href="#">umbra.lib.BytecodeStringsMnemonics</a> (The container for all the byte code mnemonic strings) . . . . .	190
<a href="#">umbra.editor.actions.BytecodeSynchrAction</a> (This class defines action of the synchronisation for a byte code position with an appropriate point in the source code) . . . . .	197
<a href="#">umbra.instructions.BytecodeTextParser</a> (This class handles the operations which are common to all the document parsers that are used in Umbra) . . . . .	201
<a href="#">umbra.editor.parsing.BytecodeWhitespaceDetector</a> (This class defines objects that are able to chceck if a particular character is a whitespace) . . . . .	210
<a href="#">umbra.editor.parsing.BytecodeWordDetector</a> (The class implements the way the words are scanned in the Eclipse scanners used in the Umbra <a href="#">editor</a> ) . . . . .	212
<a href="#">umbra.instructions.CannotCallRuleException</a> (This class is used to mark possible errors in quick dispatcher automaton <a href="#">DispatchingAutomaton</a> ) . . . . .	214
<a href="#">umbra.editor.actions.history.ClearHistoryAction</a> (The bytecode <a href="#">editor</a> action that removes all the historical versions of code) . . . . .	215
<a href="#">umbra.editor.parsing.ColorManager</a> (This object manages the allocation and deallocation of the system colors that are used in the colouring in the bytecode editors) . . . . .	217
<a href="#">umbra.editor.ColorModeContainer</a> (This class is a static container that keeps the value of current colouring style that is obtained after each refreshing (which takes place when a byte code document is created too)) . . . . .	220
<a href="#">umbra.editor.parsing.ColorValues</a> (The interface defining colours used in particular colouring styles) . . . . .	223
<a href="#">umbra.instructions.ast.CommentLineController</a> (This class handles the creation and correctness of line controllers that form comments) . . . . .	228
<a href="#">umbra.java.actions.CommitAction</a> (The action is used to commit changes made to Java source code) . . . . .	231
<a href="#">umbra.instructions.ast.ConversionInstruction</a> (This class handles the creation and correctness for the <a href="#">instructions</a> with no parameters which convert types) . . . . .	234
<a href="#">umbra.java.actions.DisasBCEL</a> (This class defines the action related to Java source <a href="#">editor</a> ) . . . . .	240
<a href="#">umbra.instructions.DispatchingAutomaton</a> (This class implements an automaton which is used to quickly determine the type of the currently analysed line of the byte code text) . . . . .	245

<a href="#">umbra.editor.DocumentSynchroniser</a> (This class handles the logic of the synchronisation of the cursor positions between the source code and the byte code documents ) . . . . .	247
<a href="#">umbra.lib.EclipseIdentifiers</a> (This is just a container for textual Eclipse identifiers of objects defined in Umbra ) . . . . .	251
<a href="#">umbra.instructions.ast.EmptyLineController</a> (This is a class for a line with whitespaces only) . . . . .	253
<a href="#">umbra.instructions.ast.FieldInstruction</a> (This class handles the creation and correctness for <a href="#">instructions</a> to store and load field values ) . . . . .	255
<a href="#">umbra.lib.FileNames</a> (This is just container for operations on the file names used in the Umbra plugin (i.e) . . . . .	259
<a href="#">umbra.instructions.FragmentParser</a> (This class is used to parse fragments of byte code textual documents ) . . . . .	269
<a href="#">umbra.lib.GUIMessages</a> (This is just container for texts of all the GUI messages ) . . . . .	271
<a href="#">umbra.instructions.ast.HeaderLineController</a> (This is a class for lines in bytecode files that occur at the beginning of methods ) . . . . .	279
<a href="#">umbra.editor.actions.history.HistoryAction</a> (This class defines an action that adds current byte code snapshot to the <a href="#">history</a> stack ) . . . . .	282
<a href="#">umbra.lib.HistoryOperations</a> (This class implements the operations on history items ) . . . . .	284
<a href="#">umbra.instructions.ast.IConstInstruction</a> (This class handles the creation and correctness for <a href="#">iconst instructions</a> with no parameters ) . . . . .	291
<a href="#">umbra.instructions.ast.IincInstruction</a> (This class handles the creation and correctness for <a href="#">iinc</a> instruction ) . . . . .	295
<a href="#">umbra.instructions.InitParser</a> (This class handles the initial parsing of a byte code textual document ) . . . . .	300
<a href="#">umbra.editor.actions.info.InstalInfoAction</a> (The class implements the behaviour in case the Install Info button in the bytecode <a href="#">editor</a> is pressed ) . . . . .	306
<a href="#">umbra.instructions.ast.InstructionLineController</a> (This class defines a structure that describes a single byte code instruction and contains related BCEL structures ) . . . . .	308
<a href="#">umbra.instructions.InstructionNameParser</a> (This class is the part of the byte code instruction parser which contributes the parsing of various identifiers i.e) . . . . .	318
<a href="#">umbra.instructions.InstructionParser</a> (This class allows to parse the line with instruction ) . . . . .	324
<a href="#">umbra.instructions.InstructionParserGeneric</a> (This class is the initial part of the byte code instruction parser class ) . . . . .	328
<a href="#">umbra.instructions.InstructionParserHelper</a> (This class contains helper methods that allow check the classes of various syntactical classes occurring in Java byte code files ) . . . . .	333
<a href="#">umbra.instructions.InstructionTypeParser</a> (This class is the part of the byte code instruction parser which contributes the parsing of various type representations ) . . . . .	340
<a href="#">umbra.instructions.ast.InvokeInstruction</a> (This class handles the creation and correctness for <a href="#">invoke instructions</a> ) . . . . .	345
<a href="#">umbra.instructions.ast.JumpInstruction</a> (This class handles the creation and correctness for jump instructions ) . . . . .	349
<a href="#">umbra.instructions.ast.LdcInstruction</a> (This class is related to some subset of <a href="#">instructions</a> depending on parameters ) . . . . .	359
<a href="#">umbra.instructions.LineContext</a> (The line parser on which the parsing of the byte code textual representation is based is context sensitive ) . . . . .	363
<a href="#">umbra.instructions.ast.LoadStoreArrayInstruction</a> (This class handles the creation and correctness for certain <a href="#">instructions</a> with no parameters ) . . . . .	372
<a href="#">umbra.instructions.ast.LoadStoreConstInstruction</a> (This class handles the creation and correctness for <a href="#">instructions</a> with no parameters that perform loading and storing data on/from the operand stack ) . . . . .	381
<a href="#">umbra.instructions.ast.MultiInstruction</a> (This is abstract class for all <a href="#">instructions</a> with at least one parameter ) . . . . .	389
<a href="#">umbra.instructions.ast.NewInstruction</a> (This class handles the creation and correctness for <a href="#">instructions</a> to create objects, check their types, and cast them, namely: ) . . . . .	393

<a href="#">umbra.editor.parsing.NonRuleBasedDamagerRepairer</a> (This class is responsible for colouring these areas in a byte code <a href="#">editor</a> window which are inside one-line areas ) . . . . .	397
<a href="#">umbra.instructions.ast.NumInstruction</a> (This is abstract class for all <a href="#">instructions</a> with a number as a parameter ) . . . . .	402
<a href="#">umbra.instructions.ast.OtherInstruction</a> (This is abstract class for all <a href="#">instructions</a> which are correct with number parameter as well as with a string one (in "") ) . . . . .	406
<a href="#">umbra.instructions.Parsing</a> (This class handles the parsing of document lines ) . . . . .	408
<a href="#">umbra.instructions.ast.PushInstruction</a> (This class handles the creation and correctness for push instructions i.e ) . . . . .	413
<a href="#">umbra.instructions.ast.SingleInstruction</a> (This class handles the creation and correctness for certain <a href="#">instructions</a> with no parameters ) . . . . .	417
<a href="#">umbra.editor.parsing.SpecialNumberRule</a> (The text styling rule which extends the <a href="#">NumberRule</a> so that it allows an additional mark before (and optionally after) the number to be read (used with '#' and ") ) . . . . .	425
<a href="#">umbra.instructions.ast.StackInstruction</a> (This class handles the creation and correctness for load and store <a href="#">instructions</a> with parameters i.e ) . . . . .	428
<a href="#">umbra.instructions.ast.StringInstruction</a> (This is abstract class for all <a href="#">instructions</a> with a string as a parameter ) . . . . .	436
<a href="#">umbra.java.actions.SynchrSBAAction</a> (This class defines an action of synchronization positions form source code to bytecode ) . . . . .	438
<a href="#">umbra.instructions.ast.ThrowsLineController</a> (This is a class for a special bytecode lines related to thrown exceptions, not to be edited by a user ) . . . . .	442
<a href="#">umbra.editor.parsing.TokenGetter</a> (This is an intermediary class which creates the Eclipse <a href="#">parsing</a> and text partitioning classes with the properties established using the Umbra colouring modes ) . . . . .	444
<a href="#">umbra.lib.UmbraClassException</a> (This is an exception used to trace situations when a problem with class file is encountered ) . . . . .	447
<a href="#">umbra.lib.UmbraException</a> (This is an exception used in tracing internal exceptional flows within Umbra ) . . . . .	449
<a href="#">umbra.lib.UmbraLocationException</a> (This is an exception used to trace situations when the parsing exceeded the content of a document ) . . . . .	450
<a href="#">umbra.lib.UmbraMethodException</a> (This is an exception used to trace situations when the processing reached a method which does not exist in the document being processed ) . . . . .	454
<a href="#">umbra.UmbraPlugin</a> (The main plugin class to be used in the desktop ) . . . . .	456
<a href="#">umbra.lib.UmbraRangeException</a> (This is an exception used to trace situations when the parsing exceeded some size associated with an exception ) . . . . .	459
<a href="#">umbra.lib.UmbraRuntimeException</a> (This is an exception used in reporting runtime exceptional events within Umbra ) . . . . .	461
<a href="#">umbra.lib.UmbraSynchronisationException</a> (This is an exception used to trace situations when the synchronisation is attempted for a line in the source code document not in the code areas ) . . . . .	462
<a href="#">umbra.instructions.ast.UnclassifiedInstruction</a> (This is a class responsible for all lines which are regarded to be an instruction but unable to classify ) . . . . .	463
<a href="#">umbra.instructions.ast.UnknownLineController</a> (This class is responsible for all lines that we cannot classify ) . . . . .	466
<a href="#">umbra.editor.actions.info.UserGuideAction</a> (The class implements the behaviour in case the User Guide button in the byte code <a href="#">editor</a> is pressed ) . . . . .	468

# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

source/umbra/ <a href="#">UmbraPlugin.java</a> . . . . .	564
source/umbra/editor/ <a href="#">BytecodeConfiguration.java</a> . . . . .	483
source/umbra/editor/ <a href="#">BytecodeContribution.java</a> . . . . .	484
source/umbra/editor/ <a href="#">BytecodeDocument.java</a> . . . . .	485
source/umbra/editor/ <a href="#">BytecodeDocumentProvider.java</a> . . . . .	486
source/umbra/editor/ <a href="#">BytecodeEditor.java</a> . . . . .	487
source/umbra/editor/ <a href="#">BytecodeEditorContributor.java</a> . . . . .	488
source/umbra/editor/ <a href="#">ColorModeContainer.java</a> . . . . .	489
source/umbra/editor/ <a href="#">DocumentSynchroniser.java</a> . . . . .	490
source/umbra/editor/actions/ <a href="#">BytecodeColorAction.java</a> . . . . .	471
source/umbra/editor/actions/ <a href="#">BytecodeCombineAction.java</a> . . . . .	472
source/umbra/editor/actions/ <a href="#">BytecodeDoubleClickStrategy.java</a> . . . . .	473
source/umbra/editor/actions/ <a href="#">BytecodeEditorAction.java</a> . . . . .	474
source/umbra/editor/actions/ <a href="#">BytecodeRebuildAction.java</a> . . . . .	475
source/umbra/editor/actions/ <a href="#">BytecodeRefreshAction.java</a> . . . . .	476
source/umbra/editor/actions/ <a href="#">BytecodeSynchrAction.java</a> . . . . .	477
source/umbra/editor/actions/history/ <a href="#">BytecodeRestoreAction.java</a> . . . . .	478
source/umbra/editor/actions/history/ <a href="#">ClearHistoryAction.java</a> . . . . .	479
source/umbra/editor/actions/history/ <a href="#">HistoryAction.java</a> . . . . .	480
source/umbra/editor/actions/info/ <a href="#">InstalInfoAction.java</a> . . . . .	481
source/umbra/editor/actions/info/ <a href="#">UserGuideAction.java</a> . . . . .	482
source/umbra/editor/parsing/ <a href="#">BytecodeBMLSecScanner.java</a> . . . . .	491
source/umbra/editor/parsing/ <a href="#">BytecodePartitionScanner.java</a> . . . . .	492
source/umbra/editor/parsing/ <a href="#">BytecodeScanner.java</a> . . . . .	493
source/umbra/editor/parsing/ <a href="#">BytecodeWhitespaceDetector.java</a> . . . . .	494
source/umbra/editor/parsing/ <a href="#">BytecodeWordDetector.java</a> . . . . .	495
source/umbra/editor/parsing/ <a href="#">ColorManager.java</a> . . . . .	496
source/umbra/editor/parsing/ <a href="#">ColorValues.java</a> . . . . .	497
source/umbra/editor/parsing/ <a href="#">NonRuleBasedDamagerRepairer.java</a> . . . . .	498
source/umbra/editor/parsing/ <a href="#">SpecialNumberRule.java</a> . . . . .	499
source/umbra/editor/parsing/ <a href="#">TokenGetter.java</a> . . . . .	500
source/umbra/instructions/ <a href="#">BytecodeCommentParser.java</a> . . . . .	529
source/umbra/instructions/ <a href="#">BytecodeController.java</a> . . . . .	530

source/umbra/instructions/BytecodeControllerComments.java . . . . .	531
source/umbra/instructions/BytecodeControllerContainer.java . . . . .	532
source/umbra/instructions/BytecodeControllerHelper.java . . . . .	533
source/umbra/instructions/BytecodeTextParser.java . . . . .	534
source/umbra/instructions/CannotCallRuleException.java . . . . .	535
source/umbra/instructions/DispatchingAutomaton.java . . . . .	536
source/umbra/instructions/FragmentParser.java . . . . .	537
source/umbra/instructions/InitParser.java . . . . .	538
source/umbra/instructions/InstructionNameParser.java . . . . .	539
source/umbra/instructions/InstructionParser.java . . . . .	540
source/umbra/instructions/InstructionParserGeneric.java . . . . .	541
source/umbra/instructions/InstructionParserHelper.java . . . . .	542
source/umbra/instructions/InstructionTypeParser.java . . . . .	543
source/umbra/instructions/LineContext.java . . . . .	544
source/umbra/instructions/Preparsing.java . . . . .	545
source/umbra/instructions/ast/AnnotationLineController.java . . . . .	501
source/umbra/instructions/ast/ArithmeticInstruction.java . . . . .	502
source/umbra/instructions/ast/ArrayInstruction.java . . . . .	503
source/umbra/instructions/ast/BytecodeLineController.java . . . . .	504
source/umbra/instructions/ast/CommentLineController.java . . . . .	505
source/umbra/instructions/ast/ConversionInstruction.java . . . . .	506
source/umbra/instructions/ast/EmptyLineController.java . . . . .	507
source/umbra/instructions/ast/FieldInstruction.java . . . . .	508
source/umbra/instructions/ast/HeaderLineController.java . . . . .	509
source/umbra/instructions/ast/IConstInstruction.java . . . . .	510
source/umbra/instructions/ast/IincInstruction.java . . . . .	511
source/umbra/instructions/ast/InstructionLineController.java . . . . .	512
source/umbra/instructions/ast/InvokeInstruction.java . . . . .	513
source/umbra/instructions/ast/JumpInstruction.java . . . . .	514
source/umbra/instructions/ast/LdcInstruction.java . . . . .	515
source/umbra/instructions/ast/LoadStoreArrayInstruction.java . . . . .	516
source/umbra/instructions/ast/LoadStoreConstInstruction.java . . . . .	517
source/umbra/instructions/ast/MultiInstruction.java . . . . .	518
source/umbra/instructions/ast/NewInstruction.java . . . . .	519
source/umbra/instructions/ast/NumInstruction.java . . . . .	520
source/umbra/instructions/ast/OtherInstruction.java . . . . .	521
source/umbra/instructions/ast/PushInstruction.java . . . . .	522
source/umbra/instructions/ast/SingleInstruction.java . . . . .	523
source/umbra/instructions/ast/StackInstruction.java . . . . .	524
source/umbra/instructions/ast/StringInstruction.java . . . . .	525
source/umbra/instructions/ast/ThrowsLineController.java . . . . .	526
source/umbra/instructions/ast/UnclassifiedInstruction.java . . . . .	527
source/umbra/instructions/ast/UnknownLineController.java . . . . .	528
source/umbra/java/actions/CommitAction.java . . . . .	546
source/umbra/java/actions/DisasBCEL.java . . . . .	547
source/umbra/java/actions/SynchrSBAction.java . . . . .	548
source/umbra/lib/BMLParsing.java . . . . .	549
source/umbra/lib/BytecodeStrings.java . . . . .	550
source/umbra/lib/BytecodeStringsGeneric.java . . . . .	551
source/umbra/lib/BytecodeStringsMnemonics.java . . . . .	552
source/umbra/lib/EclipseIdentifiers.java . . . . .	553
source/umbra/lib/FileNames.java . . . . .	554
source/umbra/lib/GUIMessages.java . . . . .	555
source/umbra/lib/HistoryOperations.java . . . . .	556

source/umbra/lib/ <a href="#">UmbraClassException.java</a>	557
source/umbra/lib/ <a href="#">UmbraException.java</a>	558
source/umbra/lib/ <a href="#">UmbraLocationException.java</a>	559
source/umbra/lib/ <a href="#">UmbraMethodException.java</a>	560
source/umbra/lib/ <a href="#">UmbraRangeException.java</a>	561
source/umbra/lib/ <a href="#">UmbraRuntimeException.java</a>	562
source/umbra/lib/ <a href="#">UmbraSynchronisationException.java</a>	563



# Chapter 5

## Namespace Documentation

### 5.1 Package umbra

#### Classes

- class [UmbraPlugin](#)  
*The main plugin class to be used in the desktop.*

#### Packages

- package [editor](#)
- package [instructions](#)
- package [java](#)
- package [lib](#)

## 5.2 Package umbra.editor

### Classes

- class [BytecodeConfiguration](#)

*This class is used by the [BytecodeEditor](#) with the matter of double click strategy and colour versions.*

- class [BytecodeContribution](#)

*This class represents a GUI element that is contributed to the eclipse GUI by the byte code [editor](#).*

- class [BytecodeDocument](#)

*This class is an abstract model of a byte code textual document.*

- class [BytecodeDocumentProvider](#)

*This class has been modified with relation to the generated automatically to allow adding listener that is responsible for error checking.*

- class [BytecodeEditor](#)

*This is the main class that defines the byte code [editor](#).*

- class [BytecodeEditorContributor](#)

*This is managing class that adds [actions](#) to workbench menus and toolbars for a byte code [editor](#).*

- class [ColorModeContainer](#)

*This class is a static container that keeps the value of current colouring style that is obtained after each refreshing (which takes place when a byte code document is created too).*

- class [DocumentSynchroniser](#)

*This class handles the logic of the synchronisation of the cursor positions between the source code and the byte code documents.*

### Packages

- package [actions](#)
- package [parsing](#)

## 5.3 Package umbra.editor.actions

### Classes

- class [BytecodeColorAction](#)

*This class defines an action of changing the coloring style.*

- class [BytecodeCombineAction](#)

*This is an action associated with a byte code [editor](#) (an extension .btc).*

- class [BytecodeDoubleClickStrategy](#)

*This class is responsible for action that is performed after a double click event in a byte code [editor](#) window.*

- class [BytecodeEditorAction](#)

*This class defines the common operations for all the byte code [editor actions](#).*

- class [BytecodeRebuildAction](#)

*This class defines action of restoring the original version of a class file (it is saved with the name prefixed with '\_') and then generating byte code (.btc) directly from it.*

- class [BytecodeRefreshAction](#)

*This is a class defining an action: save current byte code [editor](#) window and re-generate byte code from the .class file.*

- class [BytecodeSynchrAction](#)

*This class defines action of the synchronisation for a byte code position with an appropriate point in the source code.*

### Packages

- package [history](#)
- package [info](#)

## 5.4 Package umbra.editor.actions.history

### Classes

- class [BytecodeRestoreAction](#)

*This class defines action of restoring byte code from [history](#).*

- class [ClearHistoryAction](#)

*The bytecode [editor](#) action that removes all the historical versions of code.*

- class [HistoryAction](#)

*This class defines an action that adds current byte code snapshot to the [history](#) stack.*

## 5.5 Package `umbra.editor.actions.info`

### Classes

- class [InstallInfoAction](#)

*The class implements the behaviour in case the Install Info button in the bytecode editor is pressed.*

- class [UserGuideAction](#)

*The class implements the behaviour in case the User Guide button in the byte code editor is pressed.*

## 5.6 Package umbra.editor.parsing

### Classes

- class [BytecodeBMLSecScanner](#)

*This class is responsible for colouring these texts in a byte code [editor](#) window which are inside BML annotations areas.*

- class [BytecodePartitionScanner](#)

*This class is responsible for dividing the byte code document into partitions the colouring of which is governed by different rules.*

- class [BytecodeScanner](#)

*This class is responsible for colouring these texts in a byte code [editor](#) window which are outside BML annotations areas.*

- class [BytecodeWhitespaceDetector](#)

*This class defines objects that are able to check if a particular character is a whitespace.*

- class [BytecodeWordDetector](#)

*The class implements the way the words are scanned in the Eclipse scanners used in the Umbra [editor](#).*

- class [ColorManager](#)

*This object manages the allocation and deallocation of the system colors that are used in the colouring in the bytecode editors.*

- class [ColorValues](#)

*The interface defining colours used in particular colouring styles.*

- class [NonRuleBasedDamagerRepairer](#)

*This class is responsible for colouring these areas in a byte code [editor](#) window which are inside one-line areas.*

- class [SpecialNumberRule](#)

*The text styling rule which extends the [NumberRule](#) so that it allows an additional mark before (and optionally after) the number to be read (used with '#' and '').*

- class [TokenGetter](#)

*This is an intermediary class which creates the Eclipse [parsing](#) and text partitioning classes with the properties established using the Umbra colouring modes.*

## 5.7 Package umbra.instructions

### Classes

- class [BytecodeCommentParser](#)

*This class handles the operations which are connected with the handling of the comments.*

- class [BytecodeController](#)

*This class defines some structures related to BCEL as well as to the byte code [editor](#) contents.*

- class [BytecodeControllerComments](#)

*This class contains the functionality of the [BytecodeController](#) class which is responsible for the handling of the end-of-line comments and interline comments.*

- class [BytecodeControllerContainer](#)

*This class encapsulates the internal structures of the [BytecodeController](#) and gives the internal interface to them.*

- class [BytecodeControllerHelper](#)

*This class contains various helper methods that are used in the [BytecodeController](#) class.*

- class [BytecodeTextParser](#)

*This class handles the operations which are common to all the document parsers that are used in Umbra.*

- class [CannotCallRuleException](#)

*This class is used to mark possible errors in quick dispatcher automaton [DispatchingAutomaton](#).*

- class [DispatchingAutomaton](#)

*This class implements an automaton which is used to quickly determine the type of the currently analysed line of the byte code text.*

- class [FragmentParser](#)

*This class is used to parse fragments of byte code textual documents.*

- class [InitParser](#)

*This class handles the initial parsing of a byte code textual document.*

- class [InstructionNameParser](#)

*This class is the part of the byte code instruction parser which contributes the parsing of various identifiers i.e.*

- class [InstructionParser](#)

*This class allows to parse the line with instruction.*

- class [InstructionParserGeneric](#)

*This class is the initial part of the byte code instruction parser class.*

- class [InstructionParserHelper](#)

*This class contains helper methods that allow check the classes of various syntactical classes occurring in Java byte code files.*

- class [InstructionTypeParser](#)

*This class is the part of the byte code instruction parser which contributes the parsing of various type representations.*

- class [LineContext](#)

*The line parser on which the parsing of the byte code textual representation is based is context sensitive.*

- class [Preparsing](#)

*This class handles the preparsing of document lines.*

## Packages

- package [ast](#)

## 5.8 Package umbra.instructions.ast

### Classes

- class [AnnotationLineController](#)

*This class handles the creation and correctness of line controllers that contain BML annotations.*

- class [ArithmeticInstruction](#)

*This class handles the creation and correctness for arithmetic [instructions](#) with no parameters.*

- class [ArrayInstruction](#)

*This class handles the creation and correctness for the instruction to create new arrays of primitive types (newarray).*

- class [BytecodeLineController](#)

*This is completely abstract class that contains some information useful when the line is modified or BCCEL structure is created.*

- class [CommentLineController](#)

*This class handles the creation and correctness of line controllers that form comments.*

- class [ConversionInstruction](#)

*This class handles the creation and correctness for the [instructions](#) with no parameters which convert types.*

- class [EmptyLineController](#)

*This is a class for a line with whitespaces only.*

- class [FieldInstruction](#)

*This class handles the creation and correctness for [instructions](#) to store and load field values.*

- class [HeaderLineController](#)

*This is a class for lines in bytecode files that occur at the beginning of methods.*

- class [IConstInstruction](#)

*This class handles the creation and correctness for iconst [instructions](#) with no parameters.*

- class [IincInstruction](#)

*This class handles the creation and correctness for iinc instruction.*

- class [InstructionLineController](#)

*This class defines a structure that describes a single byte code instruction and contains related BCCEL structures.*

- class [InvokeInstruction](#)

*This class handles the creation and correctness for invoke [instructions](#).*

- class [JumpInstruction](#)

*This class handles the creation and correctness for jump [instructions](#).*

- class [LdcInstruction](#)

*This class is related to some subset of [instructions](#) depending on parameters.*

- class [LoadStoreArrayInstruction](#)

*This class handles the creation and correctness for certain [instructions](#) with no parameters.*

- class [LoadStoreConstInstruction](#)

*This class handles the creation and correctness for [instructions](#) with no parameters that perform loading and storing data on/from the operand stack.*

- class [MultiInstruction](#)

*This is abstract class for all [instructions](#) with at least one parameter.*

- class [NewInstruction](#)

*This class handles the creation and correctness for [instructions](#) to create objects, check their types, and cast them, namely::*

- class [NumInstruction](#)

*This is abstract class for all [instructions](#) with a number as a parameter.*

- class [OtherInstruction](#)

*This is abstract class for all [instructions](#) which are correct with number parameter as well as with a string one (in "").*

- class [PushInstruction](#)

*This class handles the creation and correctness for push [instructions](#) i.e.*

- class [SingleInstruction](#)

*This class handles the creation and correctness for certain [instructions](#) with no parameters.*

- class [StackInstruction](#)

*This class handles the creation and correctness for load and store [instructions](#) with parameters i.e.*

- class [StringInstruction](#)

*This is abstract class for all [instructions](#) with a string as a parameter.*

- class [ThrowsLineController](#)

*This is a class for a special bytecode lines related to thrown exceptions, not to be edited by a user.*

- class [UnclassifiedInstruction](#)

*This is a class responsible for all lines which are regarded to be an instruction but unable to classify.*

- class [UnknownLineController](#)

*This class is responsible for all lines that we cannot classify.*

## **5.9 Package umbra.java**

### **Packages**

- package [actions](#)

## 5.10 Package umbra.java.actions

### Classes

- class [CommitAction](#)

*The action is used to commit changes made to Java source code.*

- class [DisasBCEL](#)

*This class defines the action related to Java source [editor](#).*

- class [SynchrSBAction](#)

*This class defines an action of synchronization positions from source code to bytecode.*

## 5.11 Package umbra.lib

### Classes

- class [BMLParsing](#)

*This class is responsible for communication with BMLLib library (except code position synchronization, that calls only stateless, static methods from BMLLib).*

- class [BytecodeStrings](#)

*The container for all the byte code and BML strings.*

- class [BytecodeStringsGeneric](#)

*The class is a container for all byte code [instructions](#) and the sequences that indicate the starts and ends of comments or BML annotation areas.*

- class [BytecodeStringsMnemonics](#)

*The container for all the byte code mnemonic strings.*

- class [EclipseIdentifiers](#)

*This is just a container for textual Eclipse identifiers of objects defined in Umbra.*

- class [FileNames](#)

*This is just container for operations on the file names used in the Umbra plugin (i.e.*

- class [GUIMessages](#)

*This is just container for texts of all the GUI messages.*

- class [HistoryOperations](#)

*This class implements the operations on history items.*

- class [UmbraClassException](#)

*This is an exception used to trace situations when a problem with class file is encountered.*

- class [UmbraException](#)

*This is an exception used in tracing internal exceptional flows within Umbra.*

- class [UmbraLocationException](#)

*This is an exception used to trace situations when the parsing exceeded the content of a document.*

- class [UmbraMethodException](#)

*This is an exception used to trace situations when the processing reached a method which does not exist in the document being processed.*

- class [UmbraRangeException](#)

*This is an exception used to trace situations when the parsing exceeded some size associated with an exception.*

- class [UmbraRuntimeException](#)

*This is an exception used in reporting runtime exceptional events within Umbra.*

- class [UmbraSynchronisationException](#)

*This is an exception used to trace situations when the synchronisation is attempted for a line in the source code document not in the code areas.*

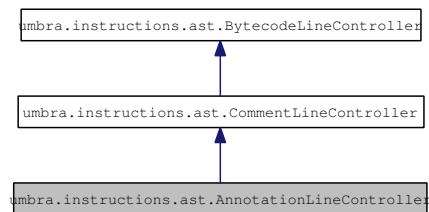
# Chapter 6

## Class Documentation

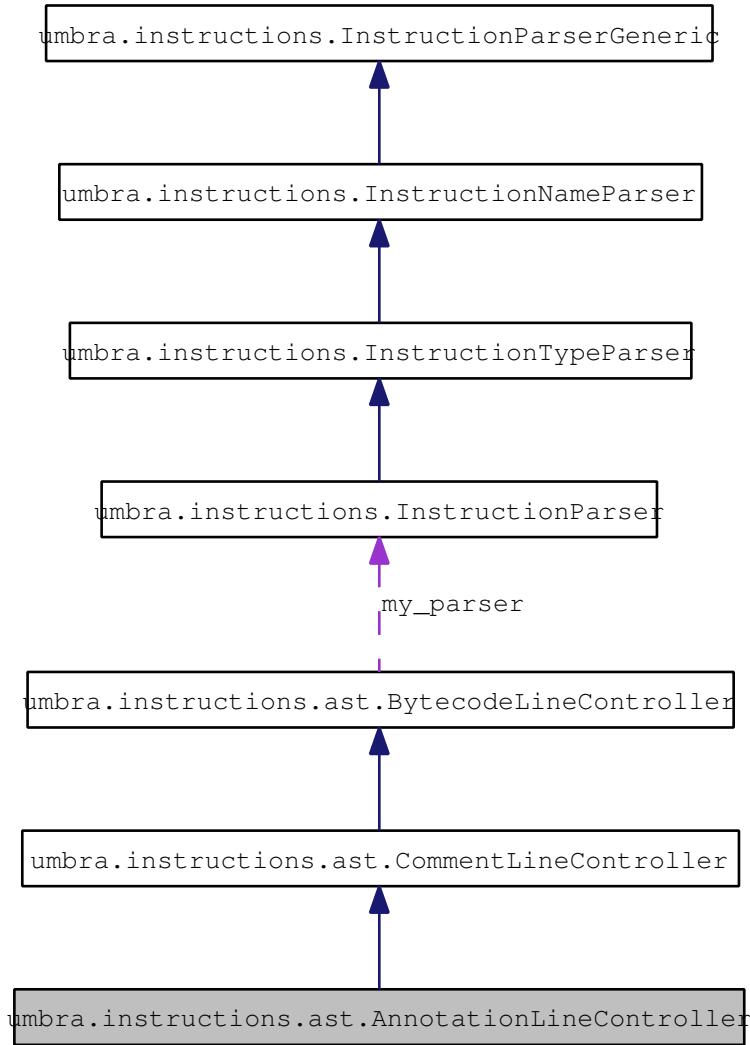
### 6.1 `umbra.instructions.ast.AnnotationLineController` Class Reference

This class handles the creation and correctness of line controllers that contain BML annotations.

Inheritance diagram for `umbra.instructions.ast.AnnotationLineController`:



Collaboration diagram for `umbra.instructions.ast.AnnotationLineController`:



## Public Member Functions

- **AnnotationLineController** (final String a\_line\_text)  
*This constructor remembers only the line text with the BML annotations.*
- final boolean **correct** ()  
*Checks the correctness of such lines.*
- boolean **isAnnotationEnd** ()  
*Checks is the line can be an end of annotation.*

## Static Public Member Functions

- static boolean **isAnnotationStart** (finalString a\_line)

The method checks if the given string can be the start of a BML annotation.

### 6.1.1 Detailed Description

This class handles the creation and correctness of line controllers that contain BML annotations.

The method number associated with the [AnnotationLineController](#) that contains the specs of a method is this method number.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 [umbra.instructions.ast.AnnotationLineController](#).AnnotationLineController (final String *a\_line\_text*)

This constructor remembers only the line text with the BML annotations.

#### Parameters:

*a\_line\_text* the string representation of the line for the line with the BML annotations

#### See also:

[BytecodeLineController](#).BytecodeLineController(String)

### 6.1.3 Member Function Documentation

#### 6.1.3.1 final boolean [umbra.instructions.ast.AnnotationLineController](#).correct ()

Checks the correctness of such lines.

The Umbra parser considers them as always correct. The actual check is done elsewhere (in BmlLib).

#### Returns:

ture

#### See also:

[BytecodeLineController](#).correct()

Reimplemented from [umbra.instructions.ast.CommentLineController](#).

### 6.1.3.2 static boolean umbra.instructions.ast.AnnotationLineController.isAnnotationStart (final String *a\_line*) [static]

The method checks if the given string can be the start of a BML annotation.

We use the heuristic that the line must start with "/\*@" possibly with some initial whitespace before the sequence.

**Parameters:**

*a\_line* the string to be checked

**Returns:**

true when the string can start annotation.

### 6.1.3.3 boolean umbra.instructions.ast.AnnotationLineController.isAnnotationEnd ()

Checks if the line can be an end of annotation.

This holds when the final non-whitespace sequence in the line is either [BytecodeStrings#ANNOT\\_LINE-END](#) or [BytecodeStrings#ANNOT\\_LINE\\_END\\_SIMPLE](#).

**Returns:**

true when the line contains the end of comment sequence, false otherwise

References [umbra.instructions.ast.BytecodeLineController.getMy\\_line\\_text\(\)](#).

Referenced by [umbra.instructions.Preparsing.getType\(\)](#).

Here is the call graph for this function:



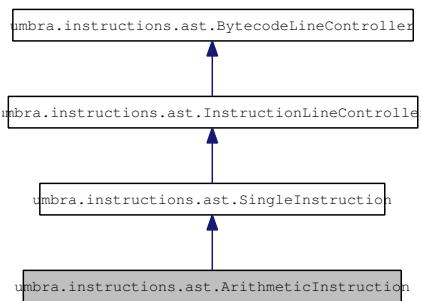
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[AnnotationLineController.java](#)

## 6.2 umbra.instructions.ast.ArithmeticInstruction Class Reference

This class handles the creation and correctness for arithmetic [instructions](#) with no parameters.

Inheritance diagram for `umbra.instructions.ast.ArithmeticInstruction`:



Collaboration diagram for `umbra.instructions.ast.ArithmeticInstruction`:



### Public Member Functions

- [ArithmeticInstruction](#) (final String a\_line\_text, final String a\_name)

*This creates an instance of an instruction named as a\_name with the line text a\_line.*

- Instruction [getInstruction](#) ()

*This method, based on the value of the mnemonic field, creates a new BCEL instruction object for an arithmetic instruction with no parameters.*

- boolean [correct](#) ()

*Simple arithmetic instruction line is correct if it has no parameter and comes from the inventory of arithmetic instructions.*

### Static Public Member Functions

- static String[] [getMnemonics](#) ()

*This method returns the array of arithmetic [instructions](#) mnemonics.*

### Private Member Functions

- Instruction [getLOpInstruction](#) (finalInstruction a\_res)

*This method creates the objects that represent [instructions](#) which perform arithmetic operations on longs.*

- Instruction [getLShiftOpInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent [instructions](#) which perform shift operations on longs.*

- Instruction [getLBoolOpInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent [instructions](#) which perform boolean operations on longs.*

- Instruction [getIOPInstruction](#) (finalInstruction a\_res)

*This method creates the objects that represent [instructions](#) which perform arithmetic operations on ints.*

- Instruction [getIBoolOpInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent [instructions](#) which perform boolean operations on ints.*

- Instruction [getFOpInstruction](#) (finalInstruction a\_res)

*This method creates the objects that represent [instructions](#) which perform arithmetic operations on floats.*

- Instruction [getDOpInstruction](#) (finalInstruction a\_res)

*This method creates the objects that represent [instructions](#) which perform arithmetic operations on doubles.*

### 6.2.1 Detailed Description

This class handles the creation and correctness for arithmetic [instructions](#) with no parameters.

The [instructions](#) handled here are in the following categories:

- arithmetic [instructions](#) for doubles,
- arithmetic [instructions](#) for floats,
- arithmetic [instructions](#) for integers, and
- arithmetic [instructions](#) for longs.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 [umbra.instructions.ast.ArithmeticInstruction.ArithmeticInstruction](#) (final String *a\_line\_text*, final String *a\_name*)

This creates an instance of an instruction named as *a\_name* with the line text *a\_line*.

Currently it just calls the constructor of the superclass.

**Parameters:**

*a\_line\_text* the line number of the instruction

*a\_name* the mnemonic name of the instruction

**See also:**

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.2.3 Member Function Documentation

#### 6.2.3.1 static String [] umbra.instructions.ast.ArithmeticInstruction.getMnemonics () [static]

This method returns the array of arithmetic [instructions](#) mnemonics.

##### Returns:

the array of the handled mnemonics

##### See also:

[InstructionLineController.getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

#### 6.2.3.2 Instruction umbra.instructions.ast.ArithmeticInstruction.getLOpInstruction (final Instruction *a\_res*) [private]

This method creates the objects that represent [instructions](#) which perform arithmetic operations on longs.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The [instructions](#) which perform arithmetic operations on longs are:

- lsub,
- ladd,
- ldiv,
- lmul,
- lneg,
- lrem,
- lcmp,
- or a bit shift operations on longs,
- or boolean operations on longs.

##### Parameters:

*a\_res* a helper value returned in case the current instruction is not in the current set

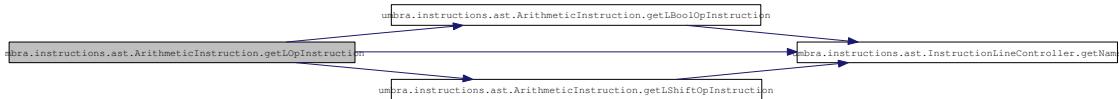
##### Returns:

the object that represents the current instruction or res in case the current instruction is not in the current set

References [umbra.instructions.ast.ArithmeticInstruction.getLBoolOpInstruction\(\)](#),  
[umbra.instructions.ast.ArithmeticInstruction.getLShiftOpInstruction\(\)](#), and [umbra.instructions.ast.InstructionLineController.getName\(\)](#).

Referenced by `umbra.instructions.ast.ArithmeticInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.2.3.3 Instruction `umbra.instructions.ast.ArithmeticInstruction.getLShiftOpInstruction` (final Instruction `a_res`) [private]

This method creates the objects that represent `instructions` which perform shift operations on longs.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The `instructions` which perform boolean operations on longs are:

- lshl,
- lshr,
- lushr.

#### Parameters:

`a_res` a helper value returned in case the current instruction is not in the current set

#### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.ArithmeticInstruction.getLOpInstruction()`.

Here is the call graph for this function:



### 6.2.3.4 Instruction `umbra.instructions.ast.ArithmeticInstruction.getLBoolOpInstruction` (final Instruction `a_res`) [private]

This method creates the objects that represent `instructions` which perform boolean operations on longs.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The `instructions` which perform boolean operations on longs are:

- land,

- lor,
- lxor.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References umbra.instructions.ast.InstructionLineController.getName().

Referenced by umbra.instructions.ast.ArithmeticInstruction.getLOpInstruction().

Here is the call graph for this function:



### 6.2.3.5 Instruction umbra.instructions.ast.ArithmeticInstruction.getIOpInstruction (final Instruction *a\_res*) [private]

This method creates the objects that represent [instructions](#) which perform arithmetic operations on ints.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The [instructions](#) which perform arithmetic operations on ints are:

- isub,
- iadd,
- idiv,
- imul,
- ineg,
- irem,
- ishl,
- iushr,
- or a boolean operation on ints.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

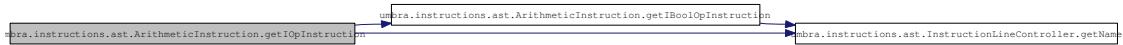
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.ArithmeticInstruction.getIBoolOpInstruction()`, and `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.ArithmeticInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.2.3.6 Instruction `umbra.instructions.ast.ArithmeticInstruction.getIBoolOpInstruction (final Instruction a_res) [private]`

This method creates the objects that represent `instructions` which perform boolean operations on ints.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The `instructions` which perform boolean operations on ints are:

- iand,
- ior,
- ixor.

#### Parameters:

`a_res` a helper value returned in case the current instruction is not in the current set

#### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.ArithmeticInstruction.getIOpInstruction()`.

Here is the call graph for this function:



### 6.2.3.7 Instruction `umbra.instructions.ast.ArithmeticInstruction.getFOpInstruction (final Instruction a_res) [private]`

This method creates the objects that represent `instructions` which perform arithmetic operations on floats.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The `instructions` which perform arithmetic operations on floats are:

- fsub,

- fadd,
- fdiv,
- fmul,
- fneg,
- frem.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.ArithmeticInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.2.3.8 Instruction `umbra.instructions.ast.ArithmeticInstruction.getDOpInstruction (final Instruction a_res) [private]`

This method creates the objects that represent `instructions` which perform arithmetic operations on doubles.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The `instructions` which perform arithmetic operations on doubles are:

- dsub,
- dadd,
- ddiv,
- dmul,
- dneg,
- drem.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.ArithmeticInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.2.3.9 Instruction `umbra.instructions.ast.ArithmeticInstruction.getInstruction()`

This method, based on the value of the mnemonic field, creates a new BCEL instruction object for an arithmetic instruction with no parameters.

The method can construct an instruction from one of the following families:

- arithmetic [instructions](#) for doubles,
- arithmetic [instructions](#) for floats,
- arithmetic [instructions](#) for integers, and
- arithmetic [instructions](#) for longs.

This method also checks the syntactical correctness of the current instruction line.

#### Returns:

the freshly constructed BCEL instruction or `null` in case the instruction is not a stack instruction and in case the instruction line is incorrect

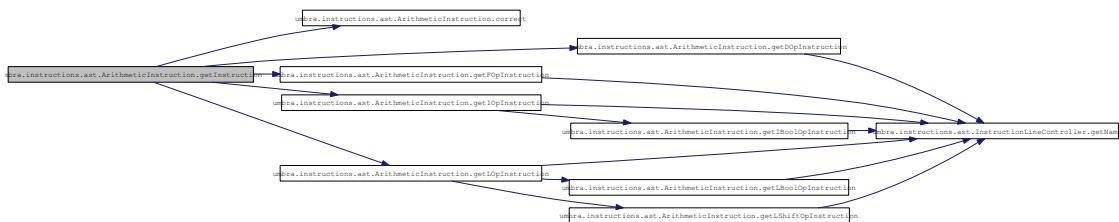
#### See also:

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

References [umbra.instructions.ast.ArithmeticInstruction.correct\(\)](#), [umbra.instructions.ast.ArithmeticInstruction.getDOpInstruction\(\)](#), [umbra.instructions.ast.ArithmeticInstruction.getFOpInstruction\(\)](#), [umbra.instructions.ast.ArithmeticInstruction.getIOpInstruction\(\)](#), [umbra.instructions.ast.ArithmeticInstruction.getLOpInstruction\(\)](#).  
and [umbra.instructions.ast.InstructionLineController.getLines\(\)](#)

Here is the call graph for this function:



### 6.2.3.10 boolean umbra.instructions.ast.ArithmeticInstruction.correct ()

Simple arithmetic instruction line is correct if it has no parameter and comes from the inventory of arithmetic [instructions](#).

**Returns:**

true when the instruction mnemonic is the only text in the line of the instruction text

**See also:**

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

Referenced by [umbra.instructions.ast.ArithmeticInstruction.getInstruction\(\)](#).

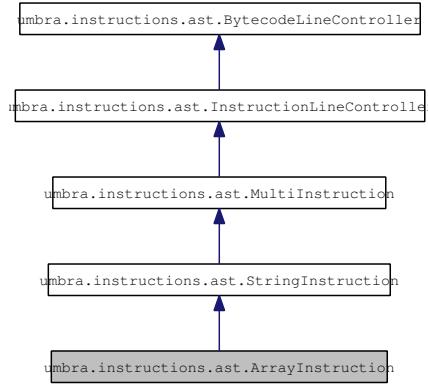
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[ArithmeticInstruction.java](#)

## 6.3 umbra.instructions.ast.ArrayInstruction Class Reference

This class handles the creation and correctness for the instruction to create new arrays of primitive types (newarray).

Inheritance diagram for `umbra.instructions.ast.ArrayInstruction`:



Collaboration diagram for `umbra.instructions.ast.ArrayInstruction`:



### Public Member Functions

- `ArrayInstruction (final String a_line_text, final String a_name)`  
*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.*
- `final Instruction getInstruction ()`  
*This method, based on the value of the mnemonic name, creates a new BCEL instruction object for a push instruction.*
- `final boolean correct ()`  
*Array instruction line is correct if it has one parameter being the type of the array elements.*

### Static Public Member Functions

- `static String[] getMnemonics ()`  
*This method returns the array of array `instructions` mnemonics.*

### Private Member Functions

- Type `getType ()`  
*This method parses the type name parameter of the current instruction.*

## Static Private Attributes

- static final Type[] **TYPES**

*The types of the bytecode types used for the creation of array [instructions](#).*

### 6.3.1 Detailed Description

This class handles the creation and correctness for the instruction to create new arrays of primitive types (newarray).

#### Author:

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 **umbra.instructions.ast.ArrayInstruction.ArrayInstruction (final String *a\_line\_text*, final String *a\_name*)**

This creates an instance of an instruction named as *a\_name* in a line the text of which is *a\_line\_text*. Currently it just calls the constructor of the superclass.

#### Parameters:

*a\_line\_text* the line number of the instruction  
*a\_name* the mnemonic name of the instruction

#### See also:

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.3.3 Member Function Documentation

#### 6.3.3.1 **static String [ ] umbra.instructions.ast.ArrayInstruction.getMnemonics () [static]**

This method returns the array of array [instructions](#) mnemonics.

#### Returns:

the array of the handled mnemonics

#### See also:

[InstructionLineController](#).[getMnemonics](#)()

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

### 6.3.3.2 Type `umbra.instructions.ast.ArrayInstruction.getType()` [private]

This method parses the type name parameter of the current instruction.

This method retrieves the type name value of the parameter of the instruction in `BytecodeLineController#getMy_line_text()`. This parameter is located after the mnemonic (with some whitespace inbetween). The method assumes `BytecodeLineController#getMy_line_text()` is correct.

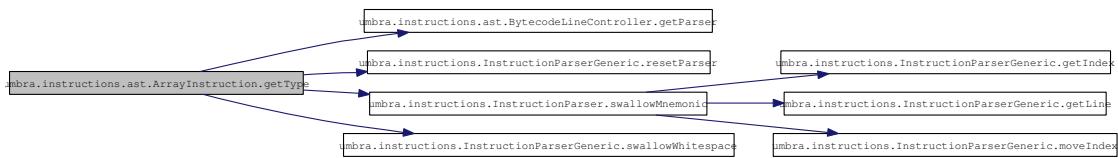
**Returns:**

the value of the type name

References `umbra.instructions.ast.BytecodeLineController.getParser()`, `umbra.instructions.InstructionParserGeneric.resetParser()`, `umbra.instructions.InstructionParser.swallowMnemonic()`, `umbra.instructions.InstructionParserGeneric.swallowWhitespace()`, and `umbra.instructions.ast.ArrayInstruction.TYPES`.

Referenced by `umbra.instructions.ast.ArrayInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.3.3.3 final Instruction `umbra.instructions.ast.ArrayInstruction.getInstruction()`

This method, based on the value of the mnemonic name, creates a new BCEL instruction object for a push instruction.

It computes the parameter of the instruction before the instruction is constructed. The method can construct one of the [instructions](#):

- newarray.

This method also checks the syntactical correctness of the current instruction line.

**Returns:**

the freshly constructed BCEL instruction or `null` in case the instruction is not a newarray instruction and in case the instruction line is incorrect

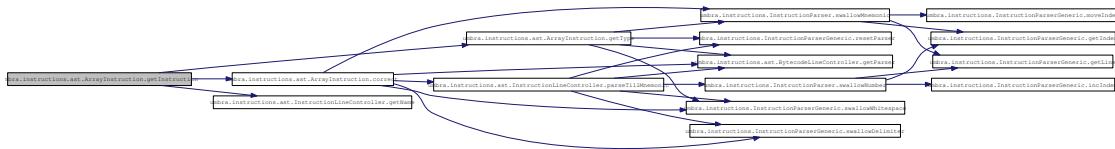
**See also:**

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References `umbra.instructions.ast.ArrayInstruction.correct()`, `umbra.instructions.ast.InstructionLineController.getName()`, and `umbra.instructions.ast.ArrayInstruction.getType()`.

Here is the call graph for this function:



#### 6.3.3.4 final boolean umbra.instructions.ast.ArrayInstruction.correct ()

Array instruction line is correct if it has one parameter being the type of the array elements.

The exact definition of this kind of a line is as follows: whitespace number : whitespace mnemonic whitespace < whitespace typename whitespace > whitespace lineend

## Returns:

true when the syntax of the instruction line is correct

#### **See also:**

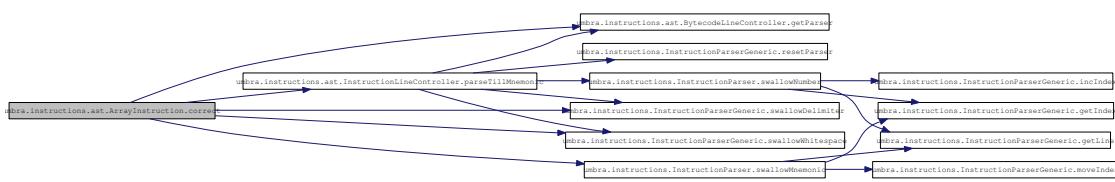
### InstructionLineController.correct()

Reimplemented from [umbra.instructions.astInstructionLineController](#).

References                  `umbra.instructions.ast.BytecodeLineController.getParser()`,                  um-  
`bra.instructions.ast.InstructionLineController.parseTillMnemonic()`,                  um-  
`bra.instructions.InstructionParserGeneric.swallowDelimiter()`, `umbra.instructions.InstructionParser.swallowMnemonic()`,  
and `umbra.instructions.InstructionParserGeneric.swallowWhitespace()`.

Referenced by `umbra.instructions.ast.ArrayInstruction.getInstruction()`.

Here is the call graph for this function:



#### **6.3.4 Member Data Documentation**

**6.3.4.1 final Type [ ] umbra.instructions.ast.ArrayInstruction.TYPES** [static, private]

## Initial value:

```
{Type.BOOLEAN, Type.CHAR, Type.FLOAT,  
Type.DOUBLE, Type.BYTE, Type.SHORT,  
Type.INT, Type.LONG}
```

The types of the bytecode types used for the creation of array [instructions](#).

It corresponds to the names in the array [BytecodeStrings#PRIMITIVE\\_TYPE\\_NAMES](#).

Referenced by [umbra.instructions.ast.ArrayInstruction.getType\(\)](#).

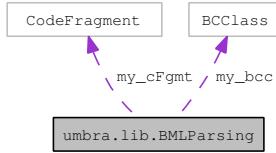
The documentation for this class was generated from the following file:

- [source/umbra/instructions/ast/ArrayInstruction.java](#)

## 6.4 umbra.lib.BMLParsing Class Reference

This class is responsible for communication with BMMLib library (except code position synchronization, that calls only stateless, static methods from BMMLib).

Collaboration diagram for umbra.lib.BMLParsing:



### Public Member Functions

- **BMLParsing** (final BCClass a\_bcc)  
*A standard constructor.*
- void **onChange** (final DocumentEvent an\_event)  
*This method should be called on every bytecode document's change.*
- BCClass **getBcc** ()
- void **setCodeString** (final String a\_code)  
*This method changes the textual representation of the byte code source.*
- boolean **isCorrect** ()  
*This method checks if the last parsed fragment is correct.*
- String **getErrorMsg** ()  
*This method return the error message for the last parsed fragment.*

### Private Attributes

- BCClass **my\_bcc**  
*This represents BML-annotated byte code whose code (if correct) is displayed in the [editor](#).*
- CodeFragment **my\_cFgmt**  
*This represents BML-annotated byte code (the same as in my\_bcc with its current (maybe incorrect) string representation and its changes since last time it was correct.*

#### 6.4.1 Detailed Description

This class is responsible for communication with BMMLib library (except code position synchronization, that calls only stateless, static methods from BMMLib).

It stores only official copies of BCClass, which represents BML-annotated bytecode. All the JavaClass' that are used in Umbra [editor](#) should be the same (==) as the one in the corresponding BCClass (accessible via [getBcc\(\).getJc\(\)](#)).

There is one [BMLParsing](#) object per one open [editor](#).

**FIXME:** make sure all the communication with BMLlib goes through this class  
<https://mobius.ucd.ie/ticket/592>

**Author:**

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))

**Version:**

a-01

## 6.4.2 Constructor & Destructor Documentation

### 6.4.2.1 [umbra.lib.BMLParsing.BMLParsing](#) (final BCClass *a\_bcc*)

A standard constructor.

Should be used just after loading a JavaClass (from file and then into BCClass).

**Parameters:**

*a\_bcc* BCClass representing bytecode in [editor](#) this object is related with. Editor's initial code should be the same as `(.equal()) bcc.printCode ()` returns.

## 6.4.3 Member Function Documentation

### 6.4.3.1 [void umbra.lib.BMLParsing.onChange](#) (final DocumentEvent *an\_event*)

This method should be called on every bytecode document's change.

It parses changes made in the document into its BCClass (if document is correct) and displays proper message (that bytecode is correct or incorrect) in the status bar.

**Parameters:**

*an\_event* -DocumentEvent describing document changes currently made, eg. event parameter of `documentChanged ()` in editor's listener.

References [umbra.lib.BMLParsing.my\\_cFgmt](#).

### 6.4.3.2 BCClass [umbra.lib.BMLParsing.getBcc](#) ()

**Returns:**

current bytecode (ast) with BML annotations. It is an official copy that all other classes related with the same [editor](#) should reference to (to make any changes in the bytecode).

References [umbra.lib.BMLParsing.my\\_bcc](#).

Referenced by [umbra.editor.BytecodeDocument.getJavaClass\(\)](#), [umbra.editor.BytecodeDocument.getMethodGen\(\)](#), [umbra.editor.BytecodeDocument.printCode\(\)](#), and [umbra.editor.BytecodeDocument.updateJavaClass\(\)](#).

**6.4.3.3 void umbra.lib.BMLParsing.setCodeString (final String *a\_code*)**

This method changes the textual representation of the byte code source.

**Parameters:**

*a\_code* the new code to associate

References umbra.lib.BMLParsing.my\_bcc, and umbra.lib.BMLParsing.my\_cFgmt.

Referenced by umbra.editor.BytocodeDocument.init().

**6.4.3.4 boolean umbra.lib.BMLParsing.isCorrect ()**

This method checks if the last parsed fragment is correct.

**Returns:**

true in case the fragment is correct, false otherwise

References umbra.lib.BMLParsing.my\_cFgmt.

Referenced by umbra.editor.BytocodeDocument.annotCorrect().

**6.4.3.5 String umbra.lib.BMLParsing.getErrorMsg ()**

This method return the error message for the last parsed fragment.

**Returns:**

the error message for the last parsed fragment

References umbra.lib.BMLParsing.my\_cFgmt.

Referenced by umbra.editor.BytocodeDocument.getAnnotError().

## 6.4.4 Member Data Documentation

**6.4.4.1 BCClass umbra.lib.BMLParsing.my\_bcc [private]**

This represents BML-annotated byte code whose code (if correct) is displayed in the [editor](#).

Referenced by umbra.lib.BMLParsing.getBcc(), and umbra.lib.BMLParsing.setCodeString().

**6.4.4.2 CodeFragment umbra.lib.BMLParsing.my\_cFgmt [private]**

This represents BML-annotated byte code (the same as in my\_bcc with its current (maybe incorrect) string representation and its changes since last time it was correct).

Referenced by umbra.lib.BMLParsing.getErrorMsg(), umbra.lib.BMLParsing.isCorrect(), umbra.lib.BMLParsing.onChange(), and umbra.lib.BMLParsing.setCodeString().

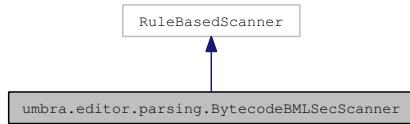
The documentation for this class was generated from the following file:

- source/umbra/lib/[BMLParsing.java](#)

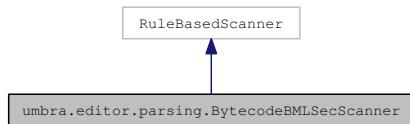
## 6.5 umbra.editor.parsing.BytecodeBMLSecScanner Class Reference

This class is responsible for colouring these texts in a byte code [editor](#) window which are inside BML annotations areas.

Inheritance diagram for `umbra.editor.parsing.BytecodeBMLSecScanner`:



Collaboration diagram for `umbra.editor.parsing.BytecodeBMLSecScanner`:



### Public Member Functions

- `BytecodeBMLSecScanner (final ColorManager the_manager, final int a_mode)`  
*The constructor initialises the scanning rules for the current scanner.*

### Private Member Functions

- `IRule createKeywordRule (final IToken a_token)`  
*This method creates a [WordRule](#) object which recognises all the BML keywords.*

### Static Private Attributes

- static final int `DOUBLE_QUOTE_RULE` = 0  
*The number of the rule that handles the recognition of the areas between the double quote characters.*
- static final int `SINGLE_QUOTE_RULE` = 1  
*The number of the rule that handles the recognition of the areas between the single quote characters.*
- static final int `WHITESPACE_RULE` = 2  
*The number of the rule that handles the recognition of the whitespace areas.*
- static final int `KEYWORD_RULE` = 3  
*The number of the rule that handles the colouring of the BML keywords.*
- static final int `NUMBER_OF_RULES` = 4

*The number of colouring rules used in this class.*

### 6.5.1 Detailed Description

This class is responsible for colouring these texts in a byte code [editor](#) window which are inside BML annotations areas.

This class uses special 4 rules which describe the way the different sequences are coloured. Colours are chosen as a token array with a particular colouring style given in the constructor.

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
 Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 **umbra.editor.parsing.BytecodeBMLSecScanner.BytecodeBMLSecScanner (final ColorManager *the\_manager*, final int *a\_mode*)**

The constructor initialises the scanning rules for the current scanner.

It creates and the scanning rules taking into account the given colour manager and colouring mode. It creates the rules to recognise strings in single and double quotes, whitespace, and BML keywords.

**Parameters:**

*the\_manager* the colour manager related to the current byte code [editor](#), it must be the same as in the current [umbra.editor.BytecodeConfiguration](#) object

*a\_mode* the number of the current colouring style, it must be the same as in the current [umbra.editor.BytecodeConfiguration](#) object

References [umbra.editor.parsing.BytecodeBMLSecScanner.createKeywordRule\(\)](#),  
[umbra.editor.parsing.BytecodeBMLSecScanner.DOUBLE\\_QUOTE\\_RULE](#),  
[umbra.editor.parsing.BytecodeBMLSecScanner.KEYWORD\\_RULE](#), [umbra.editor.parsing.BytecodeBMLSecScanner.NUMBER\\_OF\\_RULES](#), [umbra.editor.parsing.BytecodeBMLSecScanner.SINGLE\\_QUOTE\\_RULE](#), and  
[umbra.editor.parsing.BytecodeBMLSecScanner.WHITESPACE\\_RULE](#).

Here is the call graph for this function:



### 6.5.3 Member Function Documentation

#### 6.5.3.1 **IRule [umbra.editor.parsing.BytecodeBMLSecScanner.createKeywordRule \(final IToken \*a\\_token\*\) \[private\]](#)**

This method creates a [WordRule](#) object which recognises all the BML keywords.

It and assigns to them the given colour token.

**Parameters:**

*a\_token* the token to assign to the returned word rule

**Returns:**

the scanning rule that recognises the BML keywords

Referenced by `umbra.editor.parsing.BytecodeBMLSecScanner.BytecodeBMLSecScanner()`.

## 6.5.4 Member Data Documentation

### 6.5.4.1 `final int umbra.editor.parsing.BytecodeBMLSecScanner.DOUBLE_QUOTE_RULE = 0` [static, private]

The number of the rule that handles the recognition of the areas between the double quote characters.

Referenced by `umbra.editor.parsing.BytecodeBMLSecScanner.BytecodeBMLSecScanner()`.

### 6.5.4.2 `final int umbra.editor.parsing.BytecodeBMLSecScanner.SINGLE_QUOTE_RULE = 1` [static, private]

The number of the rule that handles the recognition of the areas between the single quote characters.

Referenced by `umbra.editor.parsing.BytecodeBMLSecScanner.BytecodeBMLSecScanner()`.

### 6.5.4.3 `final int umbra.editor.parsing.BytecodeBMLSecScanner.WHITESPACE_RULE = 2` [static, private]

The number of the rule that handles the recognition of the whitespace areas.

Referenced by `umbra.editor.parsing.BytecodeBMLSecScanner.BytecodeBMLSecScanner()`.

### 6.5.4.4 `final int umbra.editor.parsing.BytecodeBMLSecScanner.KEYWORD_RULE = 3` [static, private]

The number of the rule that handles the colouring of the BML keywords.

Referenced by `umbra.editor.parsing.BytecodeBMLSecScanner.BytecodeBMLSecScanner()`.

### 6.5.4.5 `final int umbra.editor.parsing.BytecodeBMLSecScanner.NUMBER_OF_RULES = 4` [static, private]

The number of colouring rules used in this class.

Referenced by `umbra.editor.parsing.BytecodeBMLSecScanner.BytecodeBMLSecScanner()`.

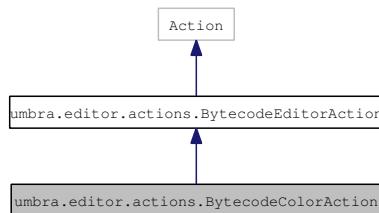
The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/[BytecodeBMLSecScanner.java](#)

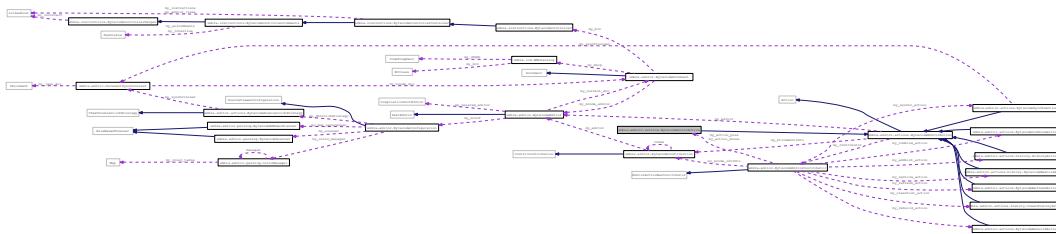
## 6.6 umbra.editor.actions.BytocodeColorAction Class Reference

This class defines an action of changing the coloring style.

Inheritance diagram for umbra.editor.actions.BytocodeColorAction:



Collaboration diagram for umbra.editor.actions.BytocodeColorAction:



### Public Member Functions

- `BytecodeColorAction (final BytecodeEditorContributor a_contr, final BytecodeContribution a_bytocode_contribution, final int a_change, final int a_mode)`

*This constructor creates the action to change the clouring mode.*

- `final void run ()`

*This method changes global value related to the coloring style and refreshes the editor window.*

- `final void setActiveEditor (final IEditorPart a_part)`

*This method sets the bytecode editor for which the action to change the colouring mode will be executed.*

### Private Attributes

- `int my_colour_delta`

*The number which decides on how the colouring mode changes (+1 for increasing, -1 for decreasing).*

- `int my_mod`

*The current colouring style, see [umbra.editor.parsing.ColorValues](#).*

### 6.6.1 Detailed Description

This class defines an action of changing the coloring style.

Two instances of the class are used - one increases the coloring style mode and the other decreased the mode.

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
 Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.6.2 Constructor & Destructor Documentation

#### 6.6.2.1 `umbra.editor.actions.BytecodeColorAction.BytecodeColorAction (final BytecodeEditorContributor a_contr, final BytecodeContribution a_bytecode_contribution, final int a_change, final int a_mode)`

This constructor creates the action to change the clouring mode.

It registers the name of the action with the text "Change color" and stores locally the object which creates all the `actions` and which contributs the `editor` GUI elements to the eclipse GUI, and the information on the color change direction (+/-1), and the current colouring mode value.

**Parameters:**

*a\_contr* the current manager that initialises `actions` for the bytecode plugin

*a\_bytecode\_contribution* the GUI elements contributed to the eclipse GUI by the byte code `editor`.  
 This reference should be the same as in the parameter *a\_contr*.

*a\_change* +1 for increasing, -1 for decreasing the colouring mode

*a\_mode* the initial colouring mode

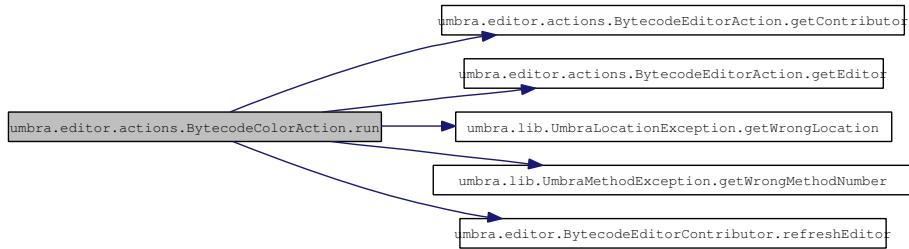
### 6.6.3 Member Function Documentation

#### 6.6.3.1 `final void umbra.editor.actions.BytecodeColorAction.run ()`

This method changes global value related to the coloring style and refreshes the `editor` window.

References `umbra.editor.actions.BytecodeEditorAction.getContributor()`, `umbra.editor.actions.BytecodeEditorAction.getEditor()`, `umbra.lib.UmbraLocationException.getWrongLocation()`, `umbra.lib.UmbraMethodException.getWrongMethodNumber()`, `umbra.editor.actions.BytecodeColorAction.my_colour_delta`, `umbra.editor.actions.BytecodeColorAction.my_mod`, and `umbra.editor.BytecodeEditorContributor.refreshEditor()`.

Here is the call graph for this function:



### 6.6.3.2 final void umbra.editor.actions.BytecodeColorAction setActiveEditor (final IEeditorPart a\_part)

This method sets the bytecode [editor](#) for which the action to change the colouring mode will be executed.

**Parameters:**

*a\_part* the bytecode [editor](#) for which the action will be executed

Reimplemented from [umbra.editor.actions.BytecodeEditorAction](#).

References [umbra.editor.actions.BytecodeColorAction.my\\_mod](#).

Referenced by [umbra.editor.BytecodeEditorContributor.setActiveEditor\(\)](#).

## 6.6.4 Member Data Documentation

### 6.6.4.1 int umbra.editor.actions.BytecodeColorAction.my\_colour\_delta [private]

The number which decides on how the colouring mode changes (+1 for increasing, -1 for decreasing).

Referenced by [umbra.editor.actions.BytecodeColorAction.run\(\)](#).

### 6.6.4.2 int umbra.editor.actions.BytecodeColorAction.my\_mod [private]

The current colouring style, see [umbra.editor.parsing.ColorValues](#).

Referenced by [umbra.editor.actions.BytecodeColorAction.run\(\)](#), [umbra.editor.actions.BytecodeColorAction.setActiveEditor\(\)](#), and [umbra.editor.actions.BytecodeColorAction.setMod\(\)](#).

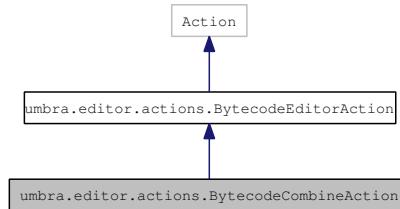
The documentation for this class was generated from the following file:

- source/umbra/editor/actions/[BytecodeColorAction.java](#)

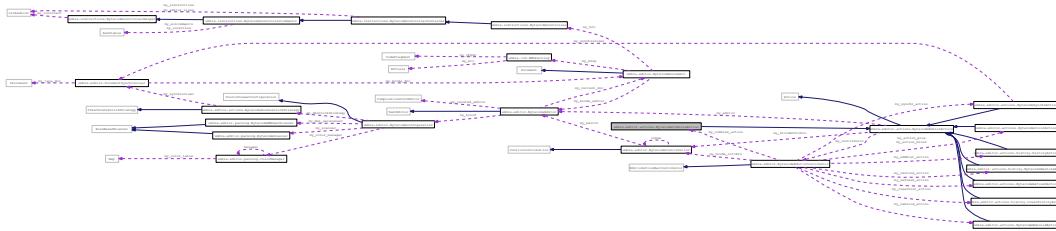
## 6.7 umbra.editor.actions.BytecodeCombineAction Class Reference

This is an action associated with a byte code [editor](#) (an extension .btc).

Inheritance diagram for `umbra.editor.actions.BytecodeCombineAction`:



Collaboration diagram for `umbra.editor.actions.BytecodeCombineAction`:



### Public Member Functions

- `BytecodeCombineAction (final BytecodeEditorContributor a_contributor, final BytecodeContributor a_bytecode_contribution)`

*This constructor creates the action to combine the byte code edits with the source code ones.*

- `final void run ()`

*This action combines the modifications that were made in the source code file with the modifications in the byte code.*

### Private Member Functions

- `void updateMethods (final IFile a_file, final IPath a_path, final String the_last_segment)`

*This method reads the Java classfile for the original Java code and replaces all the methods with the ones that were modified in the currently edited byte code file.*

- `void updateMethodsLogic (final IFile a_file, final IPath a_path, final String the_last_segment, final String a_clname, final SyntheticRepository a_repo) throws CoreException, UmbraClassException, UmbraRangeException`

*This method contains the logic of the merging together the byte code that comes from the source code manipulations and byte code manipulations.*

- `void refreshEditorWithClass (final IFile a_file, final BytecodeEditor an_editor, final JavaClass a_jc) throws UmbraClassException, PartInitException, UmbraRangeException`

*The method does the refresh operation for the current `editor` in such a way that the given file and class are associated with the edited document.*

- String `getClassPath ()`

*This method generates the classpath for the project in which the current action takes place.*

- String `classPathEntriesToString (final IClasspathEntry[ ] the_entries, final IProject a_project, final String a_project_name)`

*The method returns a string representation of a classpath the entries of which are in the parameter the\_entries and which is associated with the project a\_project.*

- ClassGen `updateModifiedMethods (final JavaClass an_old_jc, final JavaClass a_jc)`

*This method generates a new generated class representation (ClassGen) in which the methods from the class representation in the second parameter (jc) are replaced with the methods from the first parameter (oldJc) provided that my\_btcodeCntrbtn regards them as modified.*

## 6.7.1 Detailed Description

This is an action associated with a byte code `editor` (an extension .btc).

The action allows linking changes made to byte code with the ones made to the source Java code. The current implementation works only when the changes are made to different methods. In case a modification happens in the same method, the byte code modification is privileged.

### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

### Version:

a-01

## 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 umbra.editor.actions.BytocodeCombineAction.BytocodeCombineAction (final BytecodeEditorContributor a\_contributor, final BytecodeContribution a\_bytecode\_contribution)

This constructor creates the action to combine the byte code edits with the source code ones.

It registers the name of the action with the text "Combine" and stores locally the object which creates all the `actions` and which contributes the `editor` GUI elements to the eclipse GUI.

### Parameters:

`a_contributor` the manager that initialises all the `actions` within the byte code plugin

`a_bytecode_contribution` the GUI elements contributed to the eclipse GUI by the byte code `editor`.  
This reference should be the same as in the parameter `a_contributor`.

### 6.7.3 Member Function Documentation

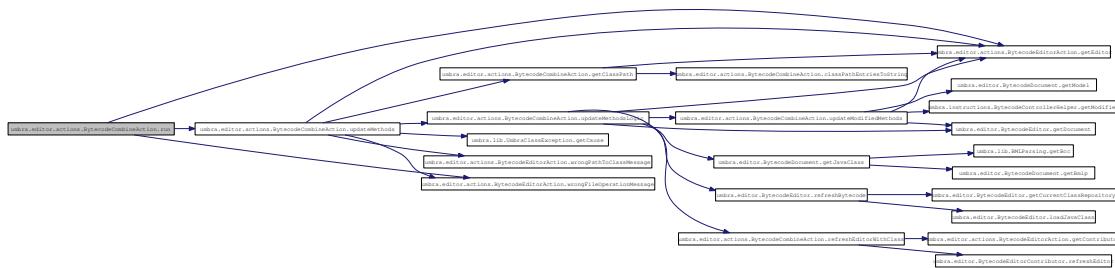
#### 6.7.3.1 final void umbra.editor.actions.BytocodeCombineAction.run ()

This action combines the modifications that were made in the source code file with the modifications in the byte code.

This method checks first if both source code and byte code files are saved. If so then it restores a clean backup copy of the class file which does not contain the changes introduced in the byte code [editor](#). Then the method reads the resulting class file and replaces all the methods with the ones that are marked as modified in the Umbra structures corresponding to the currently edited byte code file. The method does all the error handling.

References [umbra.editor.actions.BytocodeEditorAction.getEditor\(\)](#), [umbra.editor.actions.BytocodeCombineAction.updateMethods\(\)](#), and [umbra.editor.actions.BytocodeEditorAction.wrongFileOperationMessage\(\)](#).

Here is the call graph for this function:



#### 6.7.3.2 void umbra.editor.actions.BytocodeCombineAction.updateMethods (final IFile a\_file, final IPath a\_path, final String the\_last\_segment) [private]

This method reads the Java classfile for the original Java code and replaces all the methods with the ones that were modified in the currently edited byte code file.

It also does all the error handling.

##### Parameters:

*a\_file* a file edited currently by the byte code [editor](#)

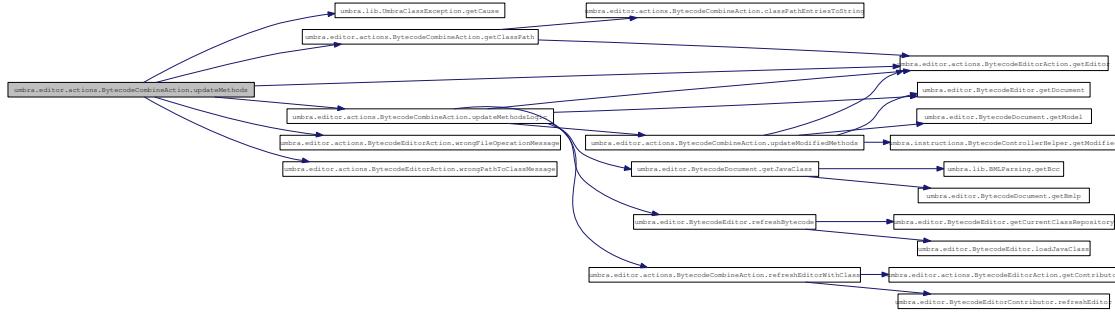
*a\_path* a workspace relative path to a Java source code file

*the\_last\_segment* the string which represents the last segment of the class file file name corresponding to the file edited by the [editor](#)

References [umbra.lib.UmbraClassException.getCause\(\)](#), [umbra.editor.actions.BytocodeCombineAction.getClassPath\(\)](#), [umbra.editor.actions.BytocodeEditorAction.getEditor\(\)](#), [umbra.editor.actions.BytocodeCombineAction.updateMethodsLogic\(\)](#), [umbra.editor.actions.BytocodeEditorAction.wrongFileOperationMessage\(\)](#), and [umbra.editor.actions.BytocodeEditorAction.wrongPathToClassMessage\(\)](#).

Referenced by [umbra.editor.actions.BytocodeCombineAction.run\(\)](#).

Here is the call graph for this function:



### 6.7.3.3 void umbra.editor.actions.BytocodeCombineAction.updateMethodsLogic (final IFile a\_file, final IPath a\_path, final String the\_last\_segment, final String a\_clname, final SyntheticRepository a\_repo) throws CoreException, UmbraClassException, UmbraRangeException [private]

This method contains the logic of the merging together the byte code that comes from the source code manipulations and byte code manipulations.

The method loads the class `a_clname` from the repository described in `a_repo`. It retrieves from the `editor` internal structures the local representation of the modified class and updates the representation with the content of the class file which is the result of the source code compilation. The result of this operation is stored in the file the name of which is the path `a_path` followed by the directory separator, followed by the file name `the_last_segment`. Finally, the current byte code `editor` is refreshed with the content of the newly generated file.

#### Parameters:

- `a_file` a file resource which is associated to the document for which the current combine action is executed
- `a_path` the path to the location where the resulting file should be stored
- `the_last_segment` the last segment of a file path (a file name) to which the new content should be generated; it is a class file name
- `a_clname` the name of the class to update the content for
- `a_repo` the repository to load the class file from

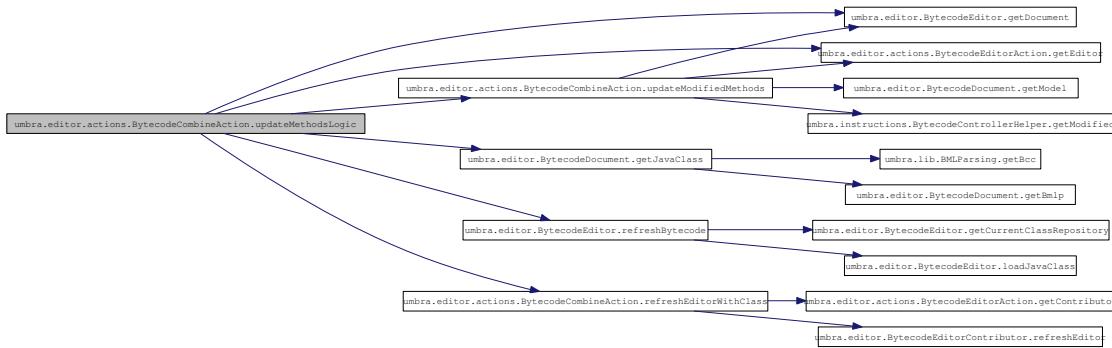
#### Exceptions:

- `CoreException` in case I/O operations on a class file failed
- `UmbraClassException` in case the class for the given name cannot be found in the given class path repository or in case the `parsing` of the BML attributes in the class file failed
- `UmbraRangeException` thrown in case a position has been reached which is outside the current document or when the textual representation has more methods than the internal one

References `umbra.editor.BytecodeEditor.getDocument()`, `umbra.editor.actions.BytocodeEditorAction.getEditor()`, `umbra.editor.BytecodeDocument.getJavaClass()`, `umbra.editor.actions.BytocodeEditorAction.my_editor`, `umbra.editor.BytecodeEditor.refreshBytecode()`, `umbra.editor.actions.BytocodeCombineAction.refreshEditorWithClass()`, and `umbra.editor.actions.BytocodeCombineAction.updateModifiedMethods()`.

Referenced by `umbra.editor.actions.BytecodeCombineAction.updateMethods()`.

Here is the call graph for this function:



#### 6.7.3.4 void `umbra.editor.actions.BytecodeCombineAction.refreshEditorWithClass (final IFile a_file, final BytecodeEditor an_editor, final JavaClass a_jc) throws UmbraClassException, PartInitException, UmbraRangeException` [private]

The method does the refresh operation for the current `editor` in such a way that the given file and class are associated with the edited document.

##### Parameters:

- `a_file` a class file to associate with the `editor`
- `an_editor` to associate the file and the class to
- `a_jc` a class file representation to associate to the `editor`

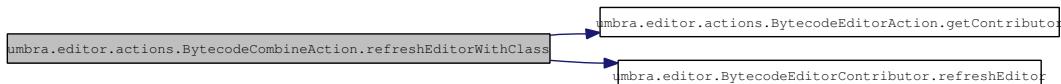
##### Exceptions:

- `UmbraClassException` in case the `parsing` of the BML attributes in the class file failed
- `PartInitException` if the new `editor` could not be created or initialised
- `UmbraRangeException` thrown in case a position has been reached which is outside the current document or when the textual representation has more methods than the internal one

References `umbra.editor.actions.BytecodeEditorAction.getContributor()`, and `umbra.editor.BytecodeEditorContributor.refreshEditor()`.

Referenced by `umbra.editor.actions.BytecodeCombineAction.updateMethodsLogic()`.

Here is the call graph for this function:



#### 6.7.3.5 String `umbra.editor.actions.BytecodeCombineAction.getClassPath ()` [private]

This method generates the classpath for the project in which the current action takes place.

In case the classpath cannot be retrieved an appropriate message is shown to the user and the classpath is set to be the empty string.

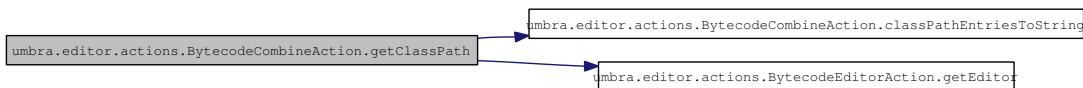
#### Returns:

the string representing the claspath

References      `umbra.editor.actions.BytecodeCombineAction.classPathEntriesToString()`,      `umbra.editor.actions.BytecodeEditorAction.getEditor()`, and `umbra.editor.actions.BytecodeEditorAction.my_editor`.

Referenced by `umbra.editor.actions.BytecodeCombineAction.updateMethods()`.

Here is the call graph for this function:



#### **6.7.3.6 String `umbra.editor.actions.BytecodeCombineAction.classPathEntriesToString` (final IClasspathEntry[ ] *the\_entries*, final IPProject *a\_project*, final String *a\_project\_name*) [private]**

The method returns a string representation of a classpath the entries of which are in the parameter *the\_entries* and which is associated with the project *a\_project*.

The *a\_project\_name* parameter is here for efficiency reasons.

#### Parameters:

- the\_entries* the entries of the classpath
- a\_project* the project with the classpath
- a\_project\_name* the name of the project with the classpath

#### Returns:

the string representation of the classpath entries

Referenced by `umbra.editor.actions.BytecodeCombineAction.getClassPath()`.

#### **6.7.3.7 ClassGen `umbra.editor.actions.BytecodeCombineAction.updateModifiedMethods` (final JavaClass *an\_old\_jc*, final JavaClass *a\_jc*) [private]**

This method generates a new generated class representation (`ClassGen`) in which the methods from the class representation in the second parameter (`jc`) are replaced with the methods from the first parameter (`oldJc`) provided that `my_btcodeCntrrbtn` regards them as modified.

#### Parameters:

- an\_old\_jc* the class from which the modifications are acquired
- a\_jc* the class for to which the modifications are added

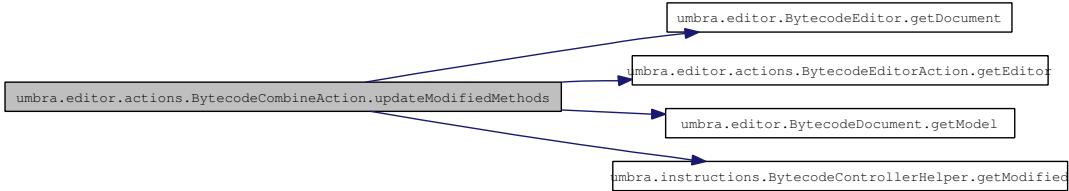
**Returns:**

the class representation with added modifications

References `umbra.editor.BytecodeEditor.getDocument()`, `umbra.editor.actions.BytecodeEditorAction.getEditor()`, `umbra.editor.BytecodeDocument.getModel()`, and `umbra.instructions.BytecodeControllerHelper.getModified()`.

Referenced by `umbra.editor.actions.BytecodeCombineAction.updateMethodsLogic()`.

Here is the call graph for this function:



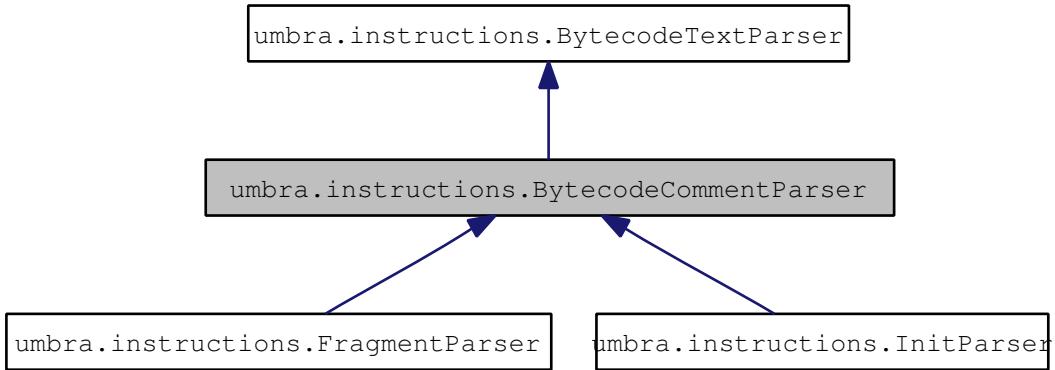
The documentation for this class was generated from the following file:

- source/umbra/editor/actions/[BytecodeCombineAction.java](#)

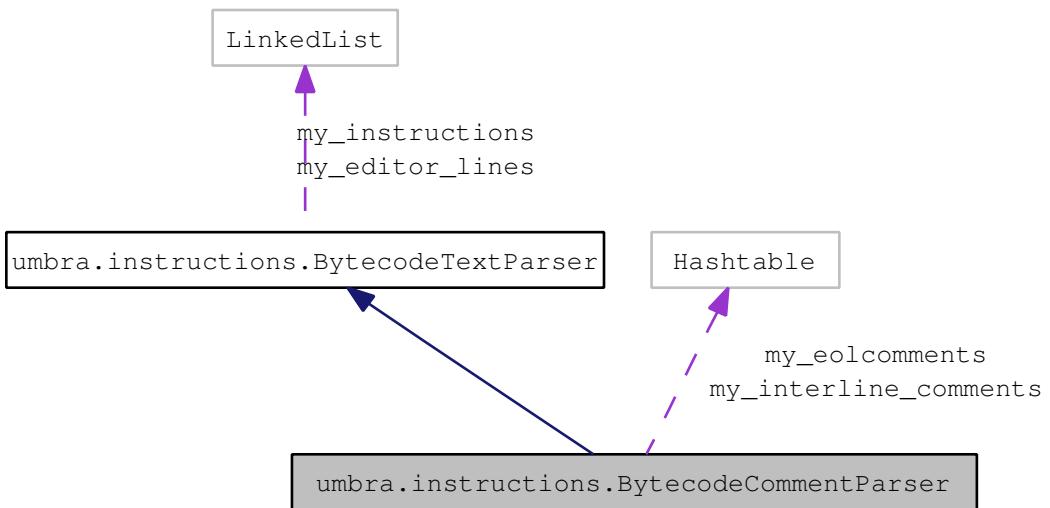
## 6.8 `umbra.instructions.BytecodeCommentParser` Class Reference

This class handles the operations which are connected with the handling of the comments.

Inheritance diagram for `umbra.instructions.BytecodeCommentParser`:



Collaboration diagram for `umbra.instructions.BytecodeCommentParser`:



### Public Member Functions

- Hashtable [getComments \(\)](#)  
*Returns the association between the lines in the internal Umbra representation and the end-of-line comments present in the textual representation.*
- Hashtable [getInterlineComments \(\)](#)  
*Returns the association between the lines in the internal Umbra representation and the multi-line comments present in the textual representation.*
- String [getCurrentComment \(\)](#)
- void [setCurrentComment \(final String a\\_comment\)](#)

---

*This method sets the value of the end-of-line comment from the currently parsed line.*

## Protected Member Functions

- [BytecodeCommentParser](#) (final String[ ] a\_comment\_array, final String[ ] a\_interline)

*This constructor initialises internal structure to represent comments.*

- String [getLineFromDoc](#) (final [BytecodeDocument](#) a\_doc, final int a\_line, final [LineContext](#) a\_ctxt) throws [UmbraLocationException](#)

*This method returns the [String](#) with the given line of the given document.*

- void [handleComments](#) (final [BytecodeLineController](#) a\_lc, final int an\_instrno)

*This method stores in the local comments structure the information about the currently extracted comment.*

- int [swallowEmptyLines](#) (final [BytecodeDocument](#) a\_doc, final int the\_current\_lno, final int the\_last\_lno, final [LineContext](#) a\_ctxt) throws [UmbraLocationException](#)

*This method parses from the given document lines which are considered to be empty lines in the given context.*

- void [clearCurrentComment](#) ()

*Clears the current representation of the multi-line comment.*

- void [addToCurrentComment](#) (final String a\_line)

*Appends the given string at the end of the current multi-line comment.*

- final void [enrichWithComment](#) (final [BytecodeLineController](#) a\_line, final int a\_pos, final int a\_instno)

*This method adds to the combination of the currently parsed text and the information from the comment structures the comment associated with the given line.*

- void [enrichWithComment](#) (final [BytecodeLineController](#) a\_line, final int a\_instno)

*This method adds to the combination of the currently parsed text and the information from the comment structures the text of the given instruction together with the comment associated with the line.*

- void [insertAt](#) (final int a\_pos, final String a\_string)

*Inserts the given string in the current representation of the combined text (class+comments) at the indicated position.*

- final int [getPosOfLine](#) (final int a\_lineno)

*Returns the position of the first character in the line of the given number.*

- String [getNewContent](#) ()

*Returns the current content of the string which contains the text of the class file combined with the comments.*

- void [adjustCommentsForInstruction](#) (final [InstructionLineController](#) a\_lc, final int an\_instrno)

*The method updates the comments structures.*

## Private Member Functions

- String [getCommentForInstr](#) (final int a\_instno)

*Retrieves the comment associated with the given instruction number.*

## Private Attributes

- String[ ] [my\\_eolcomment\\_array](#)

*This field contains the texts of end-of-line comments which were introduced in the previous session with, the current document.*

- String[ ] [my\\_interline\\_array](#)

*This field contains the texts of interline comments which were introduced in the previous session with, the current document.*

- Hashtable [my\\_eolcomments](#)

*The container of associations between the Umbra representation of lines in the byte code [editor](#) and the end-of-line comments in these lines.*

- Hashtable [my\\_interline\\_comments](#)

*The container of associations between the Umbra representation of lines in the byte code [editor](#) and the multi-line comments in these lines.*

- String [my\\_current\\_comment](#)

*This field contains the value of the end-of-line comment from the currently parsed line.*

- StringBuffer [my\\_current\\_icomment](#)

*This field contains the value of the interline comment from the currently parsed code fragment.*

- StringBuffer [my\\_combined\\_text](#)

*The combination of the currently parsed text and the information from the comment structures.*

### 6.8.1 Detailed Description

This class handles the operations which are connected with the handling of the comments.

FIXME: this is the best place to describe the logics of the comment saving  
<https://mobius.ucd.ie/ticket/560>

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

## 6.8.2 Constructor & Destructor Documentation

**6.8.2.1 umbra.instructions.BytecodeCommentParser.BytecodeCommentParser (final String[ ]  
*a\_comment\_array*, final String[ ] *a\_interline*) [protected]**

This constructor initialises internal structure to represent comments.

The given parameters are the value of the arrays which contain the comments from the previous session with the current document.

**FIXME** link to the protocol for a\_comment\_array; <https://mobius.ucd.ie/ticket/560>

## Parameters:

*a\_comment\_array* the end-of-line comments from the previous session

*a\_interline* the interline comments from the previous session

References `umbra.instructions.BytecodeCommentParser.my_combined_text`, `umbra.instructions.BytecodeCommentParser.my_eolcomment_array`, `umbra.instructions.BytecodeCommentParser.my_eolcomments`, `umbra.instructions.BytecodeCommentParser.my_interline_array`, and `umbra.instructions.BytecodeCommentParser.my_interline_comments`.

### **6.8.3 Member Function Documentation**

### 6.8.3.1 Hashtable umbra.instructions.BytecodeCommentParser.getComments ()

Returns the association between the lines in the internal Umbra representation and the end-of-line comments present in the textual representation.

## Returns:

the list of the `BytecodeLineController` objects that represent the lines with `instructions` in the currently parsed document

References `umbra.instructions.BytecodeCommentParser.my_eolcomments`.

### **6.8.3.2 Hashtable umbra.instructions.BytecodeCommentParser.getInterlineComments ()**

Returns the association between the lines in the internal Umbra representation and the multi-line comments present in the textual representation.

## Returns:

the list of the `BytecodeLineController` objects that represent the lines with `instructions` in the currently parsed document

References `umbra.instructions.BytecodeCommentParser.my_interline_comments`.

#### 6.8.3.3 String umbra.instructions.BytecodeCommentParser.getCurrentComment ()

## Returns:

the value of the current comment

References umbra.instructions.BytecodeCommentParser.my\_current\_comment.

#### 6.8.3.4 String umbra.instructions.BytecodeCommentParser.getLineFromDoc (final BytecodeDocument *a\_doc*, final int *a\_line*, final LineContext *a\_ctxt*) throws UmbraLocationException [protected]

This method returns the [String](#) with the given line of the given document.

Additionally, the method extracts the end-of-line comment and stores it in the internal state of the current object. The method needs the parsing context in case the line is a part of a multi-line context. In that case, the end-of-line comment should not be extracted.

##### Parameters:

- a\_doc* a document to extract the line from
- a\_line* the line number of the line to be extracted
- a\_ctxt* a context which indicates if we are inside a comment

##### Returns:

the string with the line content (with the end-of-line comment stripped off)

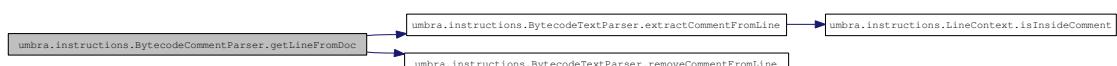
##### Exceptions:

[UmbraLocationException](#) in case the given line number is not within the given document

References [umbra.instructions.BytecodeTextParser.extractCommentFromLine\(\)](#), [umbra.instructions.BytecodeCommentParser.my\\_current\\_comment](#), and [umbra.instructions.BytecodeTextParser.removeCommentFromLine\(\)](#).

Referenced by [umbra.instructions.InitParser.swallowClassHeader\(\)](#), [umbra.instructions.BytecodeCommentParser.swallowEmptyLines\(\)](#), [umbra.instructions.InitParser.swallowMethod\(\)](#), and [umbra.instructions.InitParser.swallowMethodHeader\(\)](#).

Here is the call graph for this function:



#### 6.8.3.5 void umbra.instructions.BytecodeCommentParser.handleComments (final BytecodeLineController *a\_lc*, final int *an\_instrno*) [protected]

This method stores in the local comments structure the information about the currently extracted comment.

It also handles the enriching of the comments in the current version of the document with the information from the previous one the content of which was refreshed.

##### Parameters:

- a\_lc* the line controller to associate the comment to
- an\_instrno* the number of the instruction to be added

References [umbra.instructions.BytecodeCommentParser.my\\_current\\_comment](#), [umbra.instructions.BytecodeCommentParser.my\\_eolcomment\\_array](#), and [umbra.instructions.BytecodeCommentParser.my\\_eolcomments](#).

#### **6.8.3.6 void umbra.instructions.BytecodeCommentParser.setCurrentComment (final String a\_comment)**

This method sets the value of the end-of-line comment from the currently parsed line.

## Parameters:

*a\_comment* the current comment value to set

References `umbra.instructions.BytecodeCommentParser.my_current_comment`.

**6.8.3.7 int umbra.instructions.BytecodeCommentParser.swallowEmptyLines (final BytecodeDocument a\_doc, final int the\_current\_lno, final int the\_last\_lno, final LineContext a\_ctxt) throws UmbraLocationException [protected]**

This method parses from the given document lines which are considered to be empty lines in the given context.

A line is empty when it contains white spaces only or is one of the possible kinds of comment lines. The parsing stops at the first line which cannot be considered empty. This line will be parsed once more by the subsequent parsing procedure. We ensure here that the [AnnotationLineController](#) has the method number of either the current method or the method right after the annotation.

## Parameters:

*a\_doc* a document to extract empty lines from  
*the\_current\_ino* the first line to be analysed  
*the\_last\_ino* the line after which the document should not be analysed  
*a\_ctxt* a parsing context in which the document is analysed

## Returns:

the first line which is not an empty line; in case the end of the document is reached this is the number of lines in the document

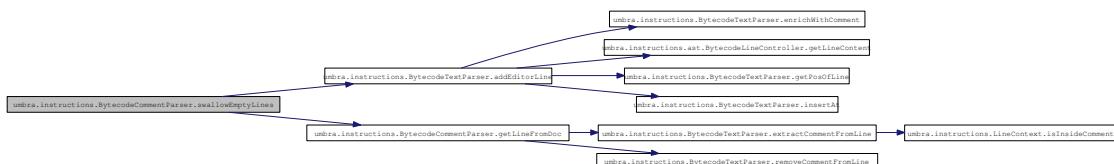
## Exceptions:

***UmbralocationException*** in case the method reaches a line number which is not within the given document

References umbra.instructions.BytecodeTextParser.addEditorLine(), and umbra.instructions.BytecodeCommentParser.getLineFromDoc().

Referenced by `umbra.instructions.InitParser.swallowClassHeader()`, and `umbra.instructions.InitParser.swallowMethod()`.

Here is the call graph for this function:



### 6.8.3.8 void **umbra.instructions.BytecodeCommentParser.clearCurrentComment ()** [protected]

Clears the current representation of the multi-line comment.

References `umbra.instructions.BytecodeCommentParser.my_current_icomment`.

### 6.8.3.9 void **umbra.instructions.BytecodeCommentParser.addToCurrentComment (final String a\_line)** [protected]

Appends the given string at the end of the current multi-line comment.

#### Parameters:

*a\_line* the string to append

References `umbra.instructions.BytecodeCommentParser.my_current_icomment`.

### 6.8.3.10 final void **umbra.instructions.BytecodeCommentParser.enrichWithComment (final BytecodeLineController a\_line, final int a\_pos, final int a\_instno)** [protected, virtual]

This method adds to the combination of the currently parsed text and the information from the comment structures the comment associated with the given line.

The method checks if the given line controller is an instruction line controller and in that case it retrieves the comment for the currently parsed line and inserts in the combined text after the text of the current instruction. We assume that the text of the instruction is already in the combined text string.

If the given line controller is not an [InstructionLineController](#) then the method does nothing.

#### Parameters:

*a\_line* a line controller to associate comments with

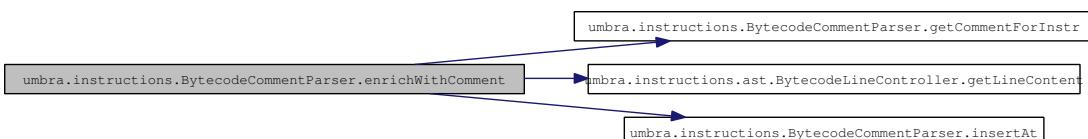
*a\_pos* the position in the combined text where the comment is to be added

*a\_instno* the number of a instruction with which the comment should be associated

Implements [umbra.instructions.BytecodeTextParser](#).

References `umbra.instructions.BytecodeCommentParser.getCommentForInstr()`, `umbra.instructions.ast.BytecodeLineController.getLineContent()`, and `umbra.instructions.BytecodeCommentParser.insertAt()`.

Here is the call graph for this function:



### 6.8.3.11 String `umbra.instructions.BytecodeCommentParser.getCommentForInstr (final int a_instno) [private]`

Retrieves the comment associated with the given instruction number.

It checks if a comment is associated with the currently parsed line. In that case this comment is returned. In case there is no comment in the parsed text, the structure of the comments from the previous session `my_eolcomment_array` is consulted.

#### Parameters:

`a_instno` the instruction number with which the comment is associated

#### Returns:

the string of the comment or `null` in case there is no comment

References `umbra.instructions.BytecodeCommentParser.my_current_comment`, and `umbra.instructions.BytecodeCommentParser.my_eolcomment_array`.

Referenced by `umbra.instructions.BytecodeCommentParser.enrichWithComment()`.

### 6.8.3.12 void `umbra.instructions.BytecodeCommentParser.enrichWithComment (final BytecodeLineController a_line, final int a_instno) [protected, virtual]`

This method adds to the combination of the currently parsed text and the information from the comment structures the text of the given instruction together with the comment associated with the line.

The method adds always the content of the line controller string and if the given line controller is an instruction line controller it retrieves the comment for the currently parsed line and inserts in the combined text after the text of the current instruction. We assume that the text of the line controller is not already in the combined text string.

If the given line controller is not an `InstructionLineController` then the method only appends the content of the given line controller

#### Parameters:

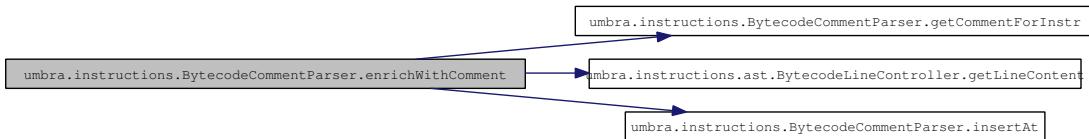
`a_line` a line controller to associate comments with

`a_instno` the number of a instruction with which the comment should be associated

Implements `umbra.instructions.BytecodeTextParser`.

References `umbra.instructions.BytecodeCommentParser.getCommentForInstr()`, `umbra.instructions.ast.BytecodeLineController.getLineContent()`, `umbra.instructions.BytecodeCommentParser.insertAt()`, and `umbra.instructions.BytecodeCommentParser.my_combined_text`.

Here is the call graph for this function:



**6.8.3.13 void umbra.instructions.BytecodeCommentParser.insertAt (final int *a\_pos*, final String *a\_string*) [protected]**

Inserts the given string in the current representation of the combined text (class+comments) at the indicated position.

The first character of the given string becomes the character at the given position and all the further characters follow. The characters of the original document starting at the given position are moved so that they start right after the inserted text.

**Parameters:**

*a\_pos* the position to insert the string at  
*a\_string* the string to insert

References umbra.instructions.BytecodeCommentParser.my\_combined\_text.

Referenced by umbra.instructions.BytecodeCommentParser.enrichWithComment().

**6.8.3.14 final int umbra.instructions.BytecodeCommentParser.getPosOfLine (final int *a\_lineno*) [protected, virtual]**

Returns the position of the first character in the line of the given number.

**Parameters:**

*a\_lineno* the number of the line to find the position for (the numbers start with 0)

**Returns:**

the position of the first character in the line

Implements [umbra.instructions.BytecodeTextParser](#).

References umbra.instructions.BytecodeCommentParser.my\_combined\_text.

**6.8.3.15 String umbra.instructions.BytecodeCommentParser.getNewContent () [protected]**

Returns the current content of the string which contains the text of the class file combined with the comments.

**Returns:**

the class text with comments

References umbra.instructions.BytecodeCommentParser.my\_combined\_text.

Referenced by umbra.instructions.InitParser.runParsing().

**6.8.3.16 void umbra.instructions.BytecodeCommentParser.adjustCommentsForInstruction (final InstructionLineController *a\_lc*, final int *an\_instrno*) [protected, virtual]**

The method updates the comments structures.

It checks if the current end-of-line comment and interline comments can be filled in with the values of the comments from the previous session and adds the association between the given line controller and the current comments.

**Parameters:**

*a\_lc* the line controller to associate the comments with

*an\_instrno* the instruction number of the given controller

Implements [umbra.instructions.BytecodeTextParser](#).

References [umbra.instructions.BytecodeCommentParser.my\\_current\\_comment](#), [umbra.instructions.BytecodeCommentParser.my\\_current\\_icomment](#), [umbra.instructions.BytecodeCommentParser.my\\_eolcomment\\_array](#), [umbra.instructions.BytecodeCommentParser.my\\_eolcomments](#), [umbra.instructions.BytecodeCommentParser.my\\_interline\\_array](#), and [umbra.instructions.BytecodeCommentParser.my\\_interline\\_comments](#).

## 6.8.4 Member Data Documentation

### 6.8.4.1 String [] [umbra.instructions.BytecodeCommentParser.my\\_eolcomment\\_array](#) [private]

This field contains the texts of end-of-line comments which were introduced in the previous session with, the current document.

The i-th entry contains the comment for the i-th instruction in the document, if this array is null then the array is not taken into account.

Referenced by [umbra.instructions.BytecodeCommentParser.adjustCommentsForInstruction\(\)](#), [umbra.instructions.BytecodeCommentParser.BytecodeCommentParser\(\)](#), [umbra.instructions.BytecodeCommentParser.getCommentForInstr\(\)](#), and [umbra.instructions.BytecodeCommentParser.handleComments\(\)](#).

### 6.8.4.2 String [] [umbra.instructions.BytecodeCommentParser.my\\_interline\\_array](#) [private]

This field contains the texts of interline comments which were introduced in the previous session with, the current document.

The i-th entry contains the comment for the i-th instruction in the document, if this array is null then the array is not taken into account.

Referenced by [umbra.instructions.BytecodeCommentParser.adjustCommentsForInstruction\(\)](#), and [umbra.instructions.BytecodeCommentParser.BytecodeCommentParser\(\)](#).

### 6.8.4.3 Hashtable [umbra.instructions.BytecodeCommentParser.my\\_eolcomments](#) [private]

The container of associations between the Umbra representation of lines in the byte code [editor](#) and the end-of-line comments in these lines.

The comments must be absent from the line representation for their correct parsing so they are held in this additional structure.

Referenced by [umbra.instructions.BytecodeCommentParser.adjustCommentsForInstruction\(\)](#), [umbra.instructions.BytecodeCommentParser.BytecodeCommentParser\(\)](#), [umbra.instructions.BytecodeCommentParser.getComments\(\)](#), and [umbra.instructions.BytecodeCommentParser.handleComments\(\)](#).

#### 6.8.4.4 Hashtable umbra.instructions.BytecodeCommentParser.my\_interline\_comments [private]

The container of associations between the Umbra representation of lines in the byte code [editor](#) and the multi-line comments in these lines.

The comments must be absent from the line representation for their correct parsing so they are held in this additional structure. [FIXME](#): this is not handled properly; <https://mobius.ucd.ie/ticket/555>

Referenced by [umbra.instructions.BytecodeCommentParser.adjustCommentsForInstruction\(\)](#), [umbra.instructions.BytecodeCommentParser.BytecodeCommentParser\(\)](#), and [umbra.instructions.BytecodeCommentParser.getInterlineComments\(\)](#).

#### 6.8.4.5 String umbra.instructions.BytecodeCommentParser.my\_current\_comment [private]

This field contains the value of the end-of-line comment from the currently parsed line.

Referenced by [umbra.instructions.BytecodeCommentParser.adjustCommentsForInstruction\(\)](#), [umbra.instructions.BytecodeCommentParser.getCommentForInstr\(\)](#), [umbra.instructions.BytecodeCommentParser.getCurrentComment\(\)](#), [umbra.instructions.BytecodeCommentParser.getLineFromDoc\(\)](#), [umbra.instructions.BytecodeCommentParser.handleComments\(\)](#), and [umbra.instructions.BytecodeCommentParser.setCurrentComment\(\)](#).

#### 6.8.4.6 StringBuffer umbra.instructions.BytecodeCommentParser.my\_current\_icomment [private]

This field contains the value of the interline comment from the currently parsed code fragment.

Referenced by [umbra.instructions.BytecodeCommentParser.addToCurrentComment\(\)](#), [umbra.instructions.BytecodeCommentParser.adjustCommentsForInstruction\(\)](#), and [umbra.instructions.BytecodeCommentParser.clearCurrentComment\(\)](#).

#### 6.8.4.7 StringBuffer umbra.instructions.BytecodeCommentParser.my\_combined\_text [private]

The combination of the currently parsed text and the information from the comment structures.

The process of parsing results in a combined version which includes both the original text and the textual representation of comments.

Referenced by [umbra.instructions.BytecodeCommentParser.BytecodeCommentParser\(\)](#), [umbra.instructions.BytecodeCommentParser.enrichWithComment\(\)](#), [umbra.instructions.BytecodeCommentParser.getNewContent\(\)](#), [umbra.instructions.BytecodeCommentParser.getPosOfLine\(\)](#), and [umbra.instructions.BytecodeCommentParser.insertAt\(\)](#).

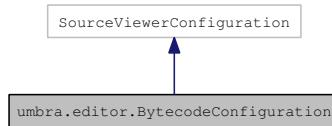
The documentation for this class was generated from the following file:

- source/umbra/instructions/[BytecodeCommentParser.java](#)

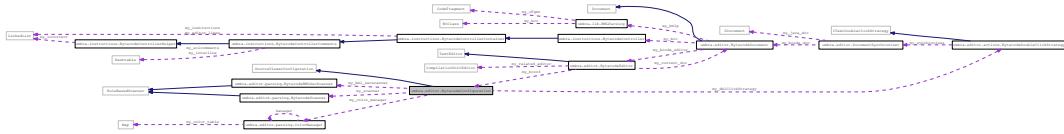
## 6.9 umbra.editor.BytecodeConfiguration Class Reference

This class is used by the [BytecodeEditor](#) with the matter of double click strategy and colour versions.

Inheritance diagram for `umbra.editor.BytecodeConfiguration`:



Collaboration diagram for `umbra.editor.BytecodeConfiguration`:



### Public Member Functions

- [BytecodeConfiguration \(\)](#)

*The constructor retrieves the current colouring mode from the [ColorModeContainer](#) and the current colour manager from [ColorManager](#).*

- final String[ ] [getConfiguredContentTypes](#) (final ISourceViewer a\_source\_viewer)  
*Returns the configured types of byte code textual document areas.*
- final ITextDoubleClickStrategy [getDoubleClickStrategy](#) (final ISourceViewer a\_source\_viewer, final String the\_content\_type)  
*This method lazily returns the value of the double click strategy associated with the current byte code [editor](#) (that means in case it is null it creates a new strategy).*
- final IPresentationReconciler [getPresentationReconciler](#) (final ISourceViewer a\_source\_viewer)  
*This method creates a new presentation reconciler ([PresentationReconciler](#)) and registers in it damagers and repairers for types ([DefaultDamagerRepairer](#)):*
- final void [disposeColor](#) ()  
*This method disposes of the operating system resources associated with the colours in the byte code [editor](#).*

### Protected Member Functions

- final [BytecodeScanner](#) [getBytecodeScanner](#) ()

*This method is a lazy getter for the scanner object.*

- final [BytecodeBMLScanner](#) [getBytecodeBMLScanner](#) ()

*This method is a lazy getter for the tag scanner object.*

## Private Attributes

- [BytocodeDoubleClickStrategy my\\_dblClickStrategy](#)

*This object handles the operation to synchronise a byte code editor point with the corresponding statement in the Java source code.*

- [BytocodeBMLSecScanner my\\_bml\\_sesscanner](#)

*The byte code scanner object used to do the colouring and text styling of the byte code areas inside of the BML areas.*

- [BytocodeScanner my\\_scanner](#)

*The byte code scanner object used to do the colouring and text styling of the byte code areas outside of the BML areas.*

- [ColorManager my\\_color\\_manager](#)

*The object which manages the allocation of the colours.*

- int [my\\_mode](#)

*The current colouring style, see [ColorValues](#).*

### 6.9.1 Detailed Description

This class is used by the [BytecodeEditor](#) with the matter of double click strategy and colour versions.

It has been generated automatically and some changes has been made, for example involving special ways of colouring and possibility of changing the colouring styles ('my\_mode' field).

#### Author:

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

Wojciech Was ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

#### Version:

a-01

### 6.9.2 Constructor & Destructor Documentation

#### 6.9.2.1 umbra.editor.BytocodeConfiguration.BytocodeConfiguration ()

The constructor retrieves the current colouring mode from the [ColorModeContainer](#) and the current colour manager from [ColorManager](#).

References [umbra.editor.parsing.ColorManager.getColorManager\(\)](#), [umbra.editor.BytocodeConfiguration.my\\_color\\_manager](#), and [umbra.editor.BytocodeConfiguration.my\\_mode](#).

Here is the call graph for this function:



### 6.9.3 Member Function Documentation

#### 6.9.3.1 final String [ ] umbra.editor.BytecodeConfiguration.getConfiguredContentTypes (final ISourceViewer *a\_source\_viewer*)

Returns the configured types of byte code textual document areas.

**Parameters:**

*a\_source\_viewer* a source viewer for which the content types are specified

**Returns:**

an array with content types for the given source viewer, in this case it contains always:

- [IDocument#DEFAULT\\_CONTENT\\_TYPE](#)
- [BytecodePartitionScanner#SECTION\\_HEAD](#)
- [BytecodePartitionScanner#SECTION\\_BML](#)

**See also:**

[SourceViewerConfiguration.getConfiguredContentTypes\(ISourceViewer\)](#)

#### 6.9.3.2 final ITextDoubleClickStrategy umbra.editor.BytecodeConfiguration.getDoubleClickStrategy (final ISourceViewer *a\_source\_viewer*, final String *the\_content\_type*)

This method lazily returns the value of the double click strategy associated with the current byte code [editor](#) (that means in case it is `null` it creates a new strategy).

**Parameters:**

*a\_source\_viewer* a source viewer for which the double click strategy is set, currently the parameter is not used

*the\_content\_type* the content type for the double click strategy

**Returns:**

the double click strategy associated with the [editor](#), the actual type is [BytecodeDoubleClickStrategy](#)

**See also:**

[SourceViewerConfiguration.getDoubleClickStrategy\(ISourceViewer, String\)](#)

References `umbra.editor.BytecodeConfiguration.my_dblClickStrategy`.

#### 6.9.3.3 final BytecodeScanner umbra.editor.BytecodeConfiguration.getBytecodeScanner () [protected]

This method is a lazy getter for the scanner object.

It checks if the corresponding field is `null`. If so it generates a new [BytecodeScanner](#) object and registers in it a default return token. This is [ColorValues#SLOT\\_DEFAULT](#).

**Returns:**

the byte code scanner object

References umbra.editor.BytecodeConfiguration.my\_color\_manager, umbra.editor.BytecodeConfiguration.my\_mode, and umbra.editor.BytecodeConfiguration.my\_scanner.

Referenced by umbra.editor.BytecodeConfiguration.getPresentationReconciler().

**6.9.3.4 final BytecodeBMLSecScanner umbra.editor.BytecodeConfiguration.getBytecodeBMLSecScanner () [protected]**

This method is a lazy getter for the tag scanner object.

It checks if the corresponding field is null. If so it generates a new [BytecodeBMLSecScanner](#) object and registers in it a default return token. This is [ColorValues#SLOT\\_TAG](#).

**Returns:**

the byte code tag scanner object

References umbra.editor.BytecodeConfiguration.my\_bml\_seccanner, umbra.editor.BytecodeConfiguration.my\_color\_manager, and umbra.editor.BytecodeConfiguration.my\_mode.

Referenced by umbra.editor.BytecodeConfiguration.getPresentationReconciler().

**6.9.3.5 final IPresentationReconciler umbra.editor.BytecodeConfiguration.getPresentationReconciler (final ISourceViewer *a\_source\_viewer*)**

This method creates a new presentation reconciler ([PresentationReconciler](#)) and registers in it damagers and repairers for types ([DefaultDamagerRepairer](#)):

- [BytecodePartitionScanner#SECTION\\_BML](#),
- [IDocument#DEFAULT\\_CONTENT\\_TYPE](#),

and for types ([NonRuleBasedDamagerRepairer](#)):

- [BytecodePartitionScanner#SECTION\\_HEAD](#),
- [BytecodePartitionScanner#SECTION\\_THROWS](#).

The [NonRuleBasedDamagerRepairer](#) is initialised with the current values of the colour manager and the mode number combined with an abstract colour indication ([ColorValues#SLOT\\_HEADER](#), [ColorValues#SLOT\\_THROWS](#)).

This method defines how the colouring works in case an edit operation is performed.

**Parameters:**

*a\_source\_viewer* the source viewer for which the reconciler is returned

**Returns:**

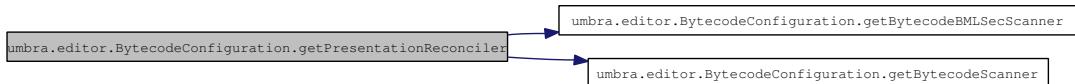
the new, configured presentation reconciler

**See also:**

`SourceViewerConfiguration.getPresentationReconciler(ISourceViewer)`

References `umbra.editor.BytecodeConfiguration.getBytecodeBMLSecScanner()`, `umbra.editor.BytecodeConfiguration.getBytecodeScanner()`, `umbra.editor.BytecodeConfiguration.my_color_manager`, and `umbra.editor.BytecodeConfiguration.my_mode`.

Here is the call graph for this function:

**6.9.3.6 final void umbra.editor.BytecodeConfiguration.disposeColor ()**

This method disposes of the operating system resources associated with the colours in the byte code `editor`.

References `umbra.editor.parsing.ColorManager.dispose()`, and `umbra.editor.BytecodeConfiguration.my_color_manager`.

Here is the call graph for this function:

**6.9.4 Member Data Documentation****6.9.4.1 BytecodeDoubleClickStrategy umbra.editor.BytecodeConfiguration.my\_dblClickStrategy [private]**

This object handles the operation to synchronise a byte code `editor` point with the corresponding statement in the Java source code.

Referenced by `umbra.editor.BytecodeConfiguration.getDoubleClickStrategy()`.

**6.9.4.2 BytecodeBMLSecScanner umbra.editor.BytecodeConfiguration.my\_bml\_secscanner [private]**

The byte code scanner object used to do the colouring and text styling of the byte code areas inside of the BML areas.

Referenced by `umbra.editor.BytecodeConfiguration.getBytecodeBMLSecScanner()`.

**6.9.4.3 BytecodeScanner umbra.editor.BytecodeConfiguration.my\_scanner [private]**

The byte code scanner object used to do the colouring and text styling of the byte code areas outside of the BML areas.

Referenced by `umbra.editor.BytecodeConfiguration.getBytecodeScanner()`.

**6.9.4.4 ColorManager umbra.editor.BytecodeConfiguration.my\_color\_manager [private]**

The object which manages the allocation of the colours.

It is shared by all the objects that handle the colouring.

Referenced by umbra.editor.BytecodeConfiguration.BytecodeConfiguration(), umbra.editor.BytecodeConfiguration.disposeColor(), umbra.editor.BytecodeConfiguration.getBytecodeBMLSecScanner(), umbra.editor.BytecodeConfiguration.getBytecodeScanner(), and umbra.editor.BytecodeConfiguration.getPresentationReconcile()

**6.9.4.5 int umbra.editor.BytecodeConfiguration.my\_mode [private]**

The current colouring style, see [ColorValues](#).

Referenced by umbra.editor.BytecodeConfiguration.BytecodeConfiguration(), umbra.editor.BytecodeConfiguration.getBytecodeBMLSecScanner(), umbra.editor.BytecodeConfiguration.getBytecodeScanner(), and umbra.editor.BytecodeConfiguration.getPresentationReconciler()

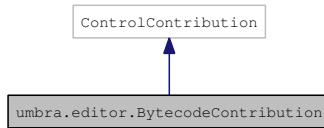
The documentation for this class was generated from the following file:

- source/umbra/editor/[BytecodeConfiguration.java](#)

## 6.10 umbra.editor.BytecodeContribution Class Reference

This class represents a GUI element that is contributed to the eclipse GUI by the byte code [editor](#).

Inheritance diagram for `umbra.editor.BytecodeContribution`:



Collaboration diagram for `umbra.editor.BytecodeContribution`:



### Public Member Functions

- `final void survive ()`  
*This method marks the current object in such a way that it cannot be replaced by a newly created one.*
- `final void addListener (final IDocument a_doc)`  
*This method adds to the document a\_doc a listener which keeps track of all the document modifications.*
- `void setActiveEditor (final IEeditorPart a_target_editor)`  
*This method sets the byte code [editor](#) for which the byte code contribution works.*

### Static Public Member Functions

- `static BytecodeContribution newItem ()`  
*The factory method which generates the `BytecodeContribution` to be used by the rest of the Umbra [editor](#).*
- `static BytecodeContribution inUse ()`  
*Returns the only contribution object that is active in the system.*

### Protected Member Functions

- `BytecodeContribution ()`  
*This creates the object and stores it in a static variable so that it is available from everywhere through `inUse()` method.*
- `final Control createControl (finalComposite a_parent)`  
*Creates the GUI control associated with the byte code [editor](#) setting a\_parent as a parent and `SWT#BORDER` as the style.*

## Private Member Functions

- void [displayCorrect \(\)](#)

*This method displays in the status line the information that something is correct.*

- void [displayError \(final String a\\_msg\)](#)

*This method displays in the status line the information about an error in the indicated line.*

## Private Attributes

- boolean [my\\_new\\_contribution\\_required](#) = true

*The flag which indicates if the current, statically cached [BytecodeContribution](#) should be replaced with a new one.*

- [BytecodeEditor my\\_editor](#)

*The current byte code [editor](#) for which the contribution works.*

## Static Private Attributes

- static [BytecodeContribution inUse](#)

*The only object of this class which is currently present in the system.*

## Classes

- class [BytecodeListener](#)

*This is a listener class that receives all the events that change the content of the current byte code document.*

### 6.10.1 Detailed Description

This class represents a GUI element that is contributed to the eclipse GUI by the byte code [editor](#).

It handles all the edit events and propagates them to the currently edited document so that they are recorded in the internal structures of the document.

The objects of this class are generated when a new [BytecodeEditor](#) is brought to life. However, in case the new [editor](#) is opened in order to refresh the content of an existing byte code document, then an old [BytecodeContribution](#) object must be reused.

FIXME: the cached object is kept in a static variable, this is probably not enough;  
<https://mobius.ucd.ie/ticket/602>

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

#### Version:

a-01

## 6.10.2 Constructor & Destructor Documentation

### 6.10.2.1 **umbra.editor.BytecodeContribution.BytecodeContribution ()** [protected]

This creates the object and stores it in a static variable so that it is available from everywhere through [inUse\(\)](#) method.

References [umbra.editor.BytecodeContribution.inUse\(\)](#).

Referenced by [umbra.editor.BytecodeContribution newItem\(\)](#).

Here is the call graph for this function:



## 6.10.3 Member Function Documentation

### 6.10.3.1 **static BytecodeContribution umbra.editor.BytecodeContribution.newItem ()** [static]

The factory method which generates the [BytecodeContribution](#) to be used by the rest of the Umbra [editor](#).

This method checks if there is a [BytecodeContribution](#) object cached and in that case it asks the object if it should be replaced by a new one. In case it should not, the currently cached object is returned. Otherwise, a new object is created.

#### Returns:

an [BytecodeContribution](#) object to be used by the system

References [umbra.editor.BytecodeContribution.BytecodeContribution\(\)](#), [umbra.editor.BytecodeContribution.inUse\(\)](#), and [umbra.editor.BytecodeContribution.my\\_new\\_contribution\\_-required](#).

Referenced by [umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor\(\)](#).

Here is the call graph for this function:



### 6.10.3.2 **static BytecodeContribution umbra.editor.BytecodeContribution.inUse ()** [static]

Returns the only contribution object that is active in the system.

#### Returns:

the active contribution object

Referenced by [umbra.editor.BytecodeContribution.BytecodeContribution\(\)](#), and [umbra.editor.BytecodeContribution newItem\(\)](#).

**6.10.3.3 final void umbra.editor.BytecodeContribution.survive ()**

This method marks the current object in such a way that it cannot be replaced by a newly created one.

References umbra.editor.BytecodeContribution.my\_new\_contribution\_required.

Referenced by umbra.editor.BytecodeEditorContributor.refreshEditor().

**6.10.3.4 final Control umbra.editor.BytecodeContribution.createControl (final Composite *a\_parent*) [protected]**

Creates the GUI control associated with the byte code `editor` setting `a_parent` as a parent and `SWT#BORDER` as the style.

It registers the current object as a data field ([Composite#setData\(Object\)](#)) in the newly created control.

**Parameters:**

`a_parent` a parent composite control under which the current control is registered

**Returns:**

the freshly created control

**See also:**

[ControlContribution.createControl\(Composite\)](#)

**6.10.3.5 void umbra.editor.BytecodeContribution.displayCorrect () [private]**

This method displays in the status line the information that something is correct.

References umbra.editor.BytecodeContribution.my\_editor.

Referenced by umbra.editor.BytecodeContribution.BytecodeListener.documentChanged().

**6.10.3.6 void umbra.editor.BytecodeContribution.displayError (final String *a\_msg*) [private]**

This method displays in the status line the information about an error in the indicated line.

**Parameters:**

`a_msg` the error message

References umbra.editor.BytecodeContribution.my\_editor.

Referenced by umbra.editor.BytecodeContribution.BytecodeListener.documentChanged().

**6.10.3.7 final void umbra.editor.BytecodeContribution.addListener (final IDocument *a\_doc*)**

This method adds to the document `a_doc` a listener which keeps track of all the document modifications.

**Parameters:**

`a_doc` a document the modifications of which will be notified by the listener

References `umbra.editor.BytecodeDocument.isListenerAdded()`.

Referenced by `umbra.editor.BytecodeDocumentProvider.createDocument()`.

Here is the call graph for this function:



#### **6.10.3.8 void umbra.editor.BytecodeContribution.setActiveEditor (final IEeditorPart a\_target\_editor)**

This method sets the byte code `editor` for which the byte code contribution works.

Currently, it does nothing as the `editor` is not used internally.

**Parameters:**

`a_target_editor` the byte code `editor` for which the action will be executed

References `umbra.editor.BytecodeContribution.my_editor`.

Referenced by `umbra.editor.BytecodeEditorContributor.setActiveEditor()`.

### **6.10.4 Member Data Documentation**

#### **6.10.4.1 BytecodeContribution umbra.editor.BytecodeContribution.inUse [static, private]**

The only object of this class which is currently present in the system.

#### **6.10.4.2 boolean umbra.editor.BytecodeContribution.my\_new\_contribution\_required = true [private]**

The flag which indicates if the current, statically cached `BytecodeContribution` should be replaced with a new one.

The default value is such that the new object should be generated.

Referenced by `umbra.editor.BytecodeContribution newItem()`, and `umbra.editor.BytecodeContribution.survive()`.

#### **6.10.4.3 BytecodeEditor umbra.editor.BytecodeContribution.my\_editor [private]**

The current byte code `editor` for which the contribution works.

Referenced by `umbra.editor.BytecodeContribution.displayCorrect()`, `umbra.editor.BytecodeContribution.displayError()`, `umbra.editor.BytecodeContribution setActiveEditor()`, and `umbra.editor.BytecodeContribution BytecodeListener.updateFragment()`.

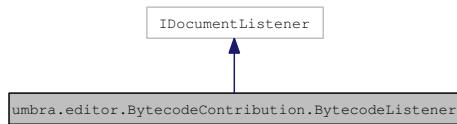
The documentation for this class was generated from the following file:

- source/umbra/editor/[BytecodeContribution.java](#)

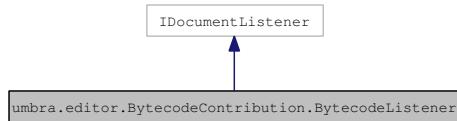
## 6.11 umbra.editor.BytecodeContribution.BytecodeListener Class Reference

This is a listener class that receives all the events that change the content of the current byte code document.

Inheritance diagram for umbra.editor.BytecodeContribution.BytecodeListener:



Collaboration diagram for umbra.editor.BytecodeContribution.BytecodeListener:



### Public Member Functions

- [BytecodeListener \(\)](#)  
*The current constructor does nothing.*
- final void [documentAboutToBeChanged](#) (final DocumentEvent an\_event)  
*This method handles the event of the change in the current byte code document.*
- final void [documentChanged](#) (final DocumentEvent an\_event)  
*This method handles the event of the change in the current byte code document.*

### Private Member Functions

- [BytecodeDocument transformDocWithMessage](#) (final IDocument a\_doc)  
*Tries to cast the given document to [BytecodeDocument](#) with appropriate message if it fails.*
- void [updateFragment](#) (final BytecodeDocument a\_doc, final int a\_start, final int an\_oldend, final int a\_newend)  
*This method handles the update of a given fragment in the given document.*
- void [messageForBadLocation](#) ()  
*Shows a pop-up with the message that the document offset is wrong.*

### Private Attributes

- int [my\\_stop\\_rem](#)

*The number of the final line which is removed from the document by the current edit operation.*

- String [my\\_oldcontent](#)

*This field contains the string representation of the document before the current change is applied.*

### 6.11.1 Detailed Description

This is a listener class that receives all the events that change the content of the current byte code document. This covers all the editing operations.

### 6.11.2 Constructor & Destructor Documentation

#### 6.11.2.1 [umbra.editor.BytecodeContribution.BytecodeListener.BytecodeListener \(\)](#)

The current constructor does nothing.

### 6.11.3 Member Function Documentation

#### 6.11.3.1 [BytecodeDocument umbra.editor.BytecodeContribution.BytecodeListener.transformDocWithMessage \(final IDocument a\\_doc\) \[private\]](#)

Tries to cast the given document to [BytecodeDocument](#) with appropriate message if it fails.

**Parameters:**

*a\_doc* a document to be cast

**Returns:**

the [BytecodeDocument](#) or null in case the cast is impossible

Referenced by [umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged\(\)](#), and [umbra.editor.BytecodeContribution.BytecodeListener.documentChanged\(\)](#).

#### 6.11.3.2 [final void umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged \(final DocumentEvent an\\_event\)](#)

This method handles the event of the change in the current byte code document.

This method is called before the textual change is made. This method initialises the [BytecodeContribution](#) object in case it has not been initialised yet.

**Parameters:**

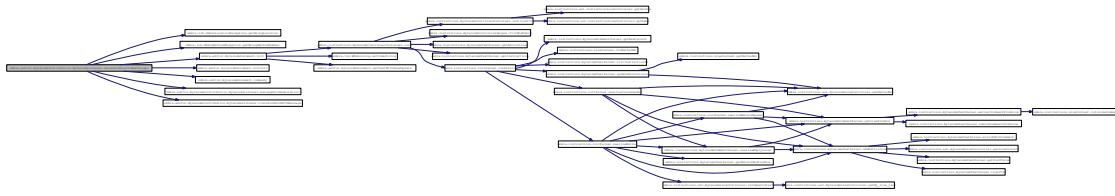
*an\_event* the event that triggers the change, it should be the same as in [documentChanged\(DocumentEvent\)](#)

**See also:**

[IDocumentListener.documentAboutToBeChanged\(DocumentEvent\)](#)

References [umbra.lib.UmbraLocationException.getWrongLocation\(\)](#), [umbra.lib.UmbraMethodException.getWrongMethodNumber\(\)](#), [umbra.editor.BytecodeDocument.init\(\)](#), [umbra.editor.BytecodeDocument.isInInit\(\)](#), [umbra.editor.BytecodeDocument.isReady\(\)](#), [umbra.editor.BytecodeContribution.BytecodeListener.messageForBadLocation\(\)](#), [umbra.editor.BytecodeContribution.BytecodeListener.my\\_oldcontent](#), [umbra.editor.BytecodeContribution.BytecodeListener.my\\_stop\\_rem](#), and [umbra.editor.BytecodeContribution.BytecodeListener.transformDocWithMessage\(\)](#).

Here is the call graph for this function:



#### **6.11.3.3 void umbra.editor.BytecodeContribution.BytecodeListener.updateFragment (final BytecodeDocument *a\_doc*, final int *a\_start*, final int *an\_oldend*, final int *a\_newend*) [private]**

This method handles the update of a given fragment in the given document.

**Parameters:**

- a\_doc*** a document which is updated, its contents are after the update
- a\_start*** the first line of the updated region
- an\_oldend*** the last line of the updated region before the update
- a\_newend*** the last line of the updated region after the update

References [umbra.lib.UmbraLocationException.getWrongLocation\(\)](#), [umbra.editor.BytecodeContribution.my\\_editor](#), [umbra.editor.BytecodeContribution.BytecodeListener.my\\_oldcontent](#), and [umbra.editor.BytecodeDocument.updateFragment\(\)](#).

Referenced by [umbra.editor.BytecodeContribution.BytecodeListener.documentChanged\(\)](#).

Here is the call graph for this function:



#### **6.11.3.4 final void umbra.editor.BytecodeContribution.BytecodeListener.documentChanged (final DocumentEvent *an\_event*)**

This method handles the event of the change in the current byte code document.

This method is called after the textual change is made. This method removes all the incorrect and correct lines in the range that has been deleted and adds all the lines in the range that has been added. Then it checks if there are errors in the resulting text and displays the information on the error.

## Parameters:

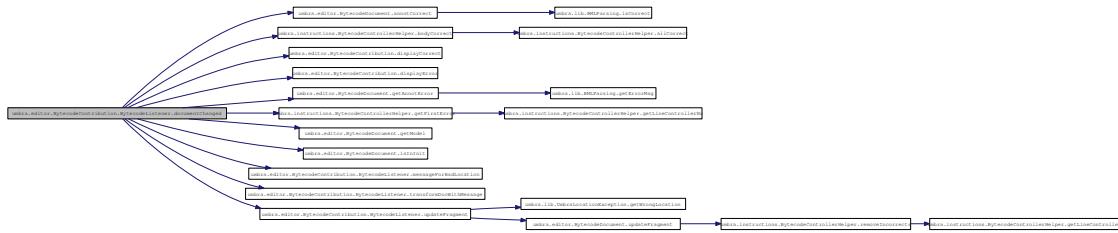
*an\_event* the event that triggers the change, it should be the same as in [documentAboutToBeChanged\(DocumentEvent\)](#)

#### **See also:**

IDocumentListener.documentChanged(DocumentEvent)

References `umbra.editor.BytecodeDocument.annotCorrect()`, `umbra.instructions.BytecodeControllerHelper.bodyCorrect()`,  
`umbra.editor.BytecodeContribution.displayCorrect()`, `umbra.editor.BytecodeContribution.displayError()`,  
`umbra.editor.BytecodeDocument.getAnnotError()`, `umbra.instructions.BytecodeControllerHelper.getFirstError()`,  
`umbra.editor.BytecodeDocument.getModel()`, `umbra.editor.BytecodeDocument.isInInit()`,  
`umbra.editor.BytecodeContribution.BytecodeListener.messageForBadLocation()`,  
`umbra.editor.BytecodeContribution.BytecodeListener.my_stop_rem`, um-  
`bra.editor.BytecodeContribution.BytecodeListener.transformDocWithMessage()`, and um-  
`bra.editor.BytecodeContribution.BytecodeListener.updateFragment()`.

Here is the call graph for this function:



#### **6.11.3.5 void umbra.editor.BytecodeContribution.BytecodeListener.messageForBadLocation () [private]**

Shows a pop-up with the message that the document offset is wrong.

Referenced by `umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged()`, and `umbra.editor.BytecodeContribution.BytecodeListener.documentChanged()`.

#### 6.11.4 Member Data Documentation

**6.11.4.1 int umbra.editor.BytocodeContribution.BytocodeListener.my\_stop\_rem [private]**

The number of the final line which is removed from the document by the current edit operation.

Note that this field must be calculated in the [documentAboutToBeChanged\(DocumentEvent\)](#) method as at that point the content to be removed is not removed yet.

Referenced by `umbra.editor.ByticodeContribution.ByticodeListener.documentAboutToBeChanged()`, and `umbra.editor.ByticodeContribution.ByticodeListener.documentChanged()`.

#### **6.11.4.2 String umbra.editor.BytecodeContribution.BytecodeListener.my\_oldcontent [private]**

This field contains the string representation of the document before the current change is applied.

Referenced by `umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged()`, and `umbra.editor.BytecodeContribution.BytecodeListener.updateFragment()`.

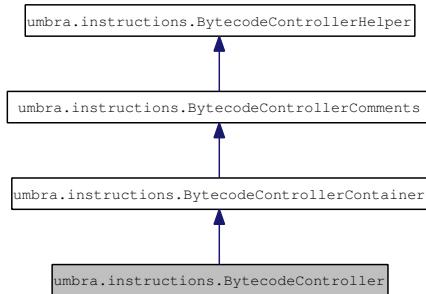
The documentation for this class was generated from the following file:

- source/umbra/editor/[BytecodeContribution.java](#)

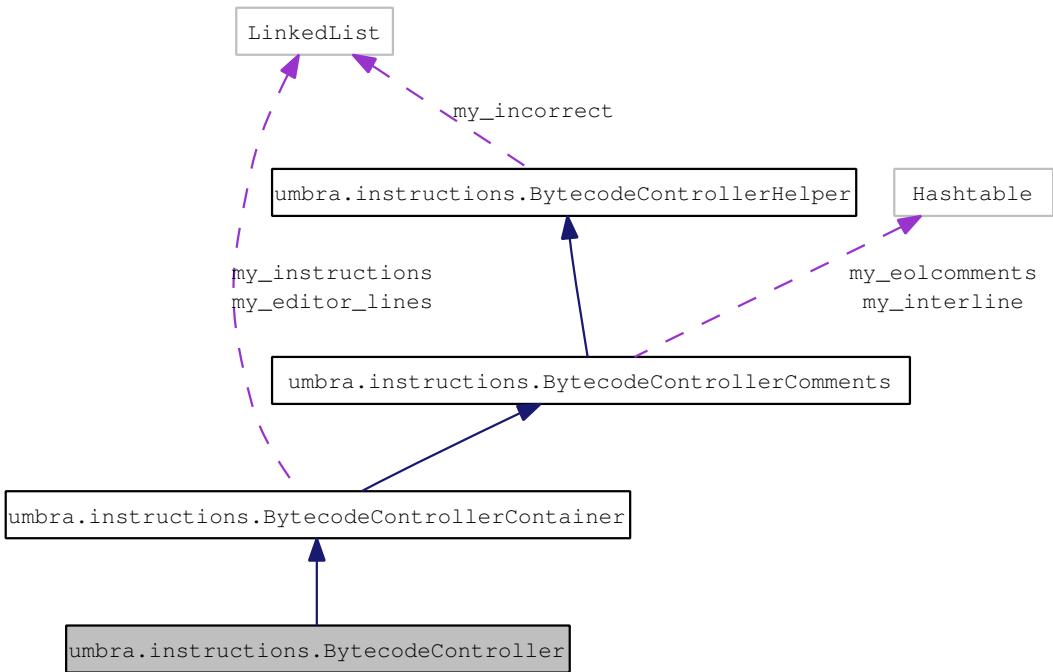
## 6.12 umbra.instructions.BytecodeController Class Reference

This class defines some structures related to BCEL as well as to the byte code [editor](#) contents.

Inheritance diagram for `umbra.instructions.BytecodeController`:



Collaboration diagram for `umbra.instructions.BytecodeController`:



### Public Member Functions

- [BytecodeController \(\)](#)

*The constructor which initialises all the internal containers to be empty.*

### Private Member Functions

- [LineContext establishCurrentContext \(final int a\\_pos\)](#)

*The method finds out which parsing context is appropriate for the given position.*

### 6.12.1 Detailed Description

This class defines some structures related to BCEL as well as to the byte code [editor](#) contents.

The structures are updated after each byte code modification and its modification allow updating BCEL. Especially a list of all lines (on purpose to check correctness) as well as a list of instruction lines to detect when BCEL modification is needed. Additional structures keep the information which method has been modified (in case of combining changes) and what comments are added to byte code.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Tomek Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([a1x@mimuw.edu.pl](mailto:a1x@mimuw.edu.pl))

#### Version:

a-01

### 6.12.2 Constructor & Destructor Documentation

#### 6.12.2.1 umbra.instructions.BytecodeController.BytecodeController ()

The constructor which initialises all the internal containers to be empty.

### 6.12.3 Member Function Documentation

#### 6.12.3.1 LineContext umbra.instructions.BytecodeController.establishCurrentContext (final int *a\_pos*) [private]

The method finds out which parsing context is appropriate for the given position.

It walks back through the structure of the [editor](#) lines until a method header is found (and in this case the context is the one appropriate for method body) or an annotation line (and in this case the context is the one appropriate for annotation).

#### Parameters:

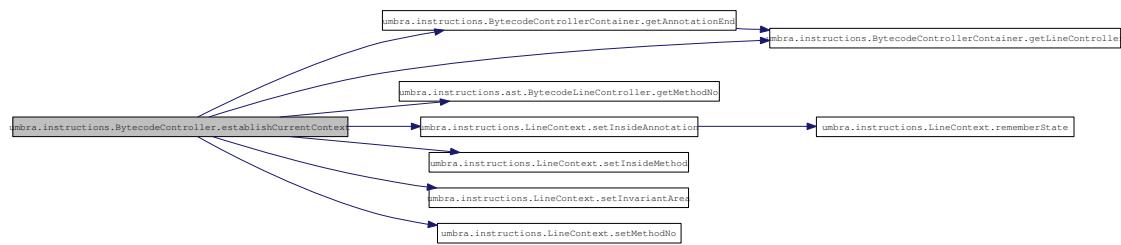
*a\_pos* a position to check the context for

#### Returns:

the context for the given position

References [umbra.instructions.BytecodeControllerContainer.getAnnotationEnd\(\)](#), [umbra.instructions.BytecodeControllerContainer.getLineController\(\)](#), [umbra.instructions.ast.BytecodeLineController.getMethodNo\(\)](#), [umbra.instructions.LineContext.setInsideAnnotation\(\)](#), [umbra.instructions.LineContext.setInsideMethod\(\)](#), [umbra.instructions.LineContext.setInvariantArea\(\)](#), and [umbra.instructions.LineContext.setMethodNo\(\)](#).

Here is the call graph for this function:



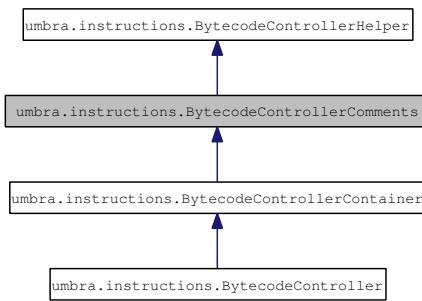
The documentation for this class was generated from the following file:

- source/umbra/instructions/[BytecodeController.java](#)

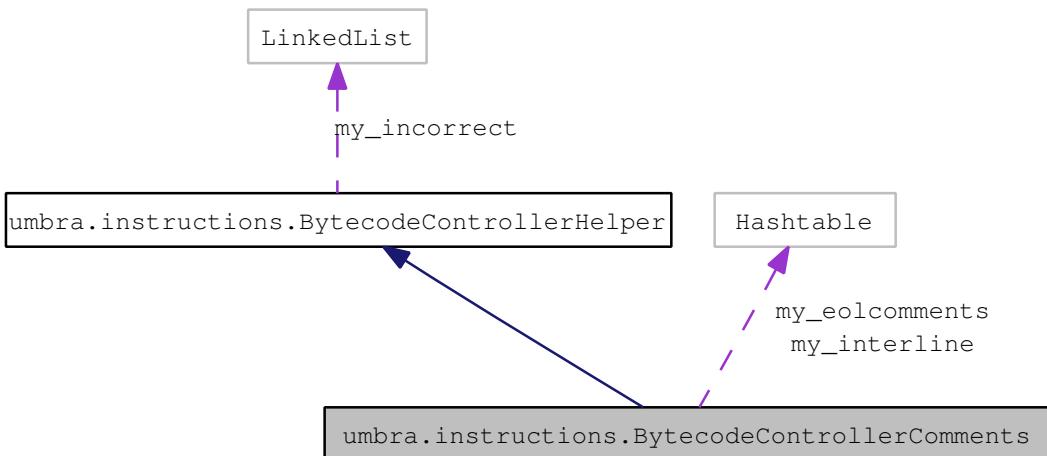
## 6.13 `umbra.instructions.BytecodeControllerComments` Class Reference

This class contains the functionality of the `BytecodeController` class which is responsible for the handling of the end-of-line comments and interline comments.

Inheritance diagram for `umbra.instructions.BytecodeControllerComments`:



Collaboration diagram for `umbra.instructions.BytecodeControllerComments`:



### Public Member Functions

- `BytecodeControllerComments ()`

*The constructor does only the initialisation of the superclass.*

### Private Attributes

- Hashtable `my_interline`

*The container of all the multi-line comments.*

- Hashtable `my_eolcomments`

*The container of associations between the Umbra representation of lines in the byte code editor and the end-of-line comments in these lines.*

### 6.13.1 Detailed Description

This class contains the functionality of the [BytecodeController](#) class which is responsible for the handling of the end-of-line comments and interline comments.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 [umbra.instructions.BytecodeControllerComments](#).BytecodeControllerComments ()

The constructor does only the initialisation of the superclass.

The fields of this class are left intact for later initialisation.

### 6.13.3 Member Data Documentation

#### 6.13.3.1 [Hashtable](#) [umbra.instructions.BytecodeControllerComments](#).my\_interline [private]

The container of all the multi-line comments.

Each element of the table is an association between an instruction line and a string with comments. The string may contain several lines of text. For a given instruction, the string contains the comment that is located after it. **FIXME:** this functionality is not realised in the current version.  
<https://mobius.ucd.ie/ticket/555>

**See also:**

[getInterlineComments\(\)](#)

#### 6.13.3.2 [Hashtable](#) [umbra.instructions.BytecodeControllerComments](#).my\_eolcomments [private]

The container of associations between the Umbra representation of lines in the byte code editor and the end-of-line comments in these lines.

The comments must be absent from the line representation for their correct parsing so they are held in this additional structure.

**See also:**

[getEOLComments\(\)](#)

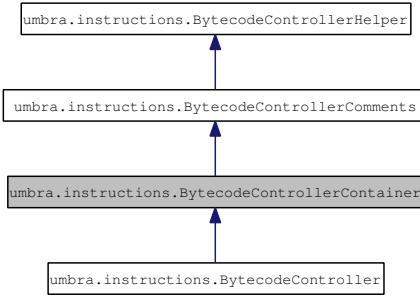
The documentation for this class was generated from the following file:

- source/umbra/instructions/[BytecodeControllerComments.java](#)

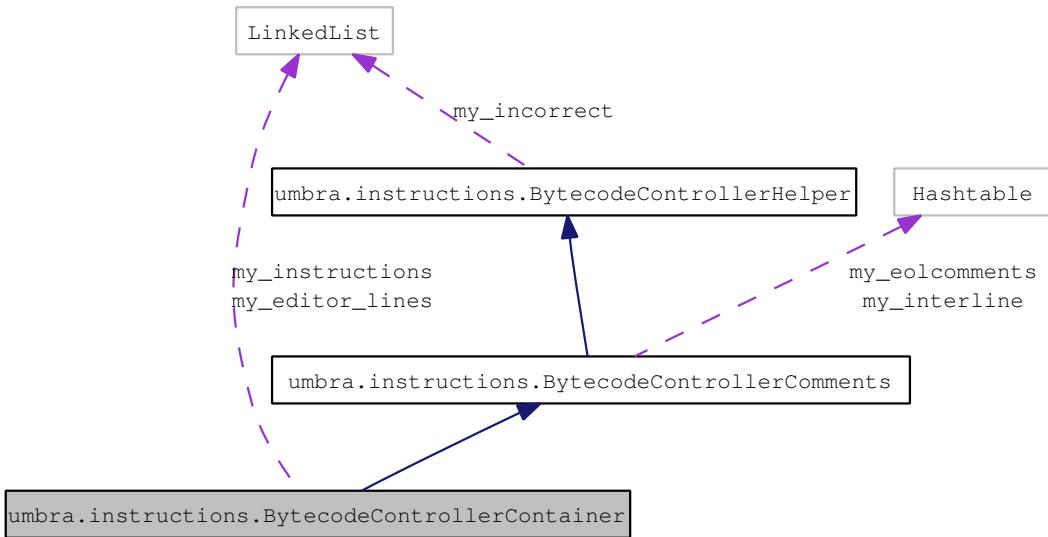
## 6.14 umbra.instructions.BytecodeControllerContainer Class Reference

This class encapsulates the internal structures of the [BytecodeController](#) and gives the internal interface to them.

Inheritance diagram for [umbra.instructions.BytecodeControllerContainer](#):



Collaboration diagram for [umbra.instructions.BytecodeControllerContainer](#):



### Public Member Functions

- [BytecodeControllerContainer \(\)](#)  
*The constructor does only the initialisation of the superclass.*
- [String init \(final BytecodeDocument a\\_doc, final String\[\] a\\_comment\\_array, final String\[\] a\\_interline\)](#) throws UmbraLocationException, UmbraMethodException  
*This method handles the initial parsing of a byte code textual document.*
- [final BytecodeLineController getLineController \(final int a\\_lineno\)](#)  
*Returns the line controller for the given line.*

## Protected Member Functions

- final int [getLineControllerNo](#) (final BytecodeLineController a\_line)  
*Returns the line number for the given line.*
- final InstructionLineController [getInstruction](#) (final int an\_insno)  
*Returns the line controller for the given instruction number.*
- final int [getNumberOfInstructions](#) ()  
*Returns the total number of the [instructions](#) in the current document.*
- final void [replaceLineController](#) (final int a\_pos, final BytecodeLineController a\_newlc)  
*Replaces the line controller at the given position with the given new line controller.*
- final int [getFirstInstructionInRegion](#) (final int the\_first, final int the\_last)  
*Finds the first instruction line controller in the given range of lines.*
- final int [getFirstInstructionAfter](#) (final int a\_pos)  
*Finds the first instruction line controller after the given point.*
- final void [appendInstructions](#) (final LinkedList the\_instructions)  
*Adds the given list of [instructions](#) at the end of the local instruction list.*
- final void [removeInstructionsInRegion](#) (final int the\_first, final int the\_last)  
*Removes from the representation of the [instructions](#) the [instructions](#) contained in the given region.*
- final void [insertInstructions](#) (final int a\_pos, final LinkedList the\_instructions)  
*Inserts the given list of [instructions](#) at the given position.*
- final void [insertEditorLine](#) (final int a\_pos, final BytecodeLineController a\_lc)  
*Inserts the given line controller at the given position.*
- final int [getAnnotationEnd](#) (final int a\_pos)  
*Returns the last annotation line for the annotation lines block starting with the given position.*
- final void [controlPrint](#) (final int an\_index)  
*This is a helper method used for debugging purposes.*

## Private Attributes

- LinkedList [my\\_editor\\_lines](#)  
*The list of all the lines in the current byte code [editor](#).*
- LinkedList [my\\_instructions](#)  
*The list of all the lines in the [editor](#) which contain codes of [instructions](#).*

### 6.14.1 Detailed Description

This class encapsulates the internal structures of the [BytecodeController](#) and gives the internal interface to them.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 [umbra.instructions.BytecodeControllerContainer.BytecodeControllerContainer\(\)](#)

The constructor does only the initialisation of the superclass.

It does no initialisation. The initialisation should be done in the [init\(BytecodeDocument, String\[\], String\[\]\)](#) method.

### 6.14.3 Member Function Documentation

#### 6.14.3.1 [String umbra.instructions.BytecodeControllerContainer.init \(final BytecodeDocument a\\_doc, final String\[\] a\\_comment\\_array, final String\[\] a\\_interline\) throws UmbraLocationException, UmbraMethodException](#)

This method handles the initial parsing of a byte code textual document.

It creates a parser [InitParser](#) and runs it with the given document and array with comments pertinent to the instruction lines. Subsequently, it initialises the internal structures to handle [editor](#) lines, [instructions](#), comments, and modifications.

**Parameters:**

*a\_doc* the byte code document with the corresponding BCEL structures linked into it

*a\_comment\_array* contains the texts of end-of-line comments, the i-th entry contains the comment for the i-th instruction in the document, if this parameter is null then the array is not taken into account

*a\_interline* contains the texts of interline comments, the i-th entry contains the comment for the i-th line in the document, if this parameter is null then the array is not taken into account FIXME: currently ignored; <https://mobius.ucd.ie/ticket/555>

**Returns:**

the string with the text of the document combined with the comments

**Exceptions:**

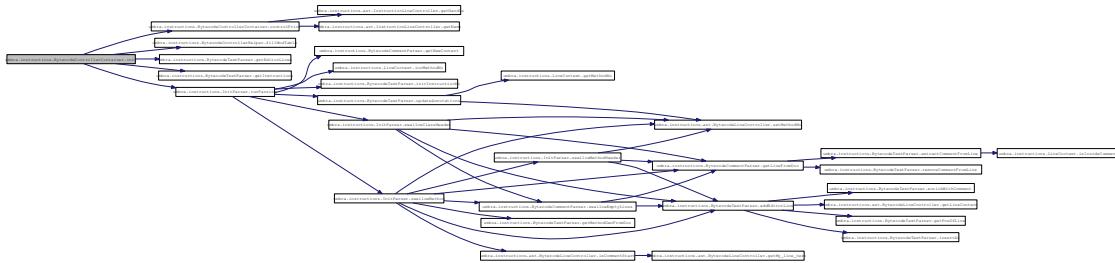
*UmbraLocationException* thrown in case a position has been reached which is outside the current document

*UmbraMethodException* thrown in case a method number has been reached which is outside the number of available methods in the document

References        [umbra.instructions.BytecodeControllerContainer.controlPrint\(\)](#),        [um-  
bra.instructions.BytecodeControllerHelper.fillModTable\(\)](#), [umbra.instructions.BytecodeTextParser.getEditorLines\(\)](#),  
[umbra.instructions.BytecodeTextParser.getInstructions\(\)](#), [umbra.instructions.BytecodeControllerContainer.my\\_](#)  
[editor\\_lines](#),        [umbra.instructions.BytecodeControllerContainer.my\\_instructions](#),        and        [um-  
bra.instructions.InitParser.runParsing\(\)](#).

Referenced by [umbra.editor.BytecodeDocument.init\(\)](#).

Here is the call graph for this function:



#### 6.14.3.2 final BytecodeLineController um- bra.instructions.BytecodeControllerContainer.getLineController (final int a\_lineno) [virtual]

Returns the line controller for the given line.

**Parameters:**

*a\_lineno* the line number of the retrieved controller line

**Returns:**

the controller line for the given line number

Implements [umbra.instructions.BytecodeControllerHelper](#).

References [umbra.instructions.BytecodeControllerContainer.my\\_editor\\_lines](#).

Referenced        by        [umbra.instructions.BytecodeControllerContainer.establishCurrentContext\(\)](#),  
[umbra.instructions.BytecodeControllerContainer.getAnnotationEnd\(\)](#),        [um-  
bra.instructions.BytecodeControllerContainer.getFirstInstructionAfter\(\)](#),        [um-  
bra.instructions.BytecodeControllerContainer.getFirstInstructionInRegion\(\)](#),        and        [um-  
bra.instructions.BytecodeControllerContainer.removeInstructionsInRegion\(\)](#).

#### 6.14.3.3 final int umbra.instructions.BytecodeControllerContainer.getLineControllerNo (final BytecodeLineController a\_line) [protected, virtual]

Returns the line number for the given line.

**Parameters:**

*a\_line* the line controller for which we obtain the number of line

**Returns:**

the number of line for the given controller or -1 if there is no such a line

Implements [umbra.instructions.BytecodeControllerHelper](#).

References [umbra.instructions.BytecodeControllerContainer.my\\_editor\\_lines](#).

**6.14.3.4 final InstructionLineController umbra.instructions.BytecodeControllerContainer.getInstruction (final int *an\_insno*) [protected]**

Returns the line controller for the given instruction number.

The instruction number is the sequence number of the instruction in the textual file.

**Parameters:**

*an\_insno* the number of the retrieved instruction

**Returns:**

the controller line for the given instruction number

References [umbra.instructions.BytecodeControllerContainer.my\\_instructions](#).

**6.14.3.5 final int umbra.instructions.BytecodeControllerContainer.getNoOfInstructions () [protected]**

Returns the total number of the [instructions](#) in the current document.

**Returns:**

the number of [instructions](#) in the current document

References [umbra.instructions.BytecodeControllerContainer.my\\_instructions](#).

**6.14.3.6 final void umbra.instructions.BytecodeControllerContainer.replaceLineController (final int *a\_pos*, final BytecodeLineController *a\_newlc*) [protected]**

Replaces the line controller at the given position with the given new line controller.

**Parameters:**

*a\_pos* the position of the line controller to be replaced

*a\_newlc* the new line controller

References [umbra.instructions.BytecodeControllerContainer.my\\_editor\\_lines](#).

**6.14.3.7 final int umbra.instructions.BytecodeControllerContainer.getFirstInstructionInRegion (final int *the\_first*, final int *the\_last*) [protected]**

Finds the first instruction line controller in the given range of lines.

**Parameters:**

*the\_first* the first line to be checked

*the\_last* the last line to be checked

**Returns:**

the number of the line with the instruction line controller or -1 in case there is no instruction line controller in the given range

References      [umbra.instructions.BytecodeControllerContainer.getLineController\(\)](#),      and  
[umbra.instructions.BytecodeControllerContainer.my\\_instructions](#).

Here is the call graph for this function:



#### 6.14.3.8 final int umbra.instructions.BytecodeControllerContainer.getFirstInstructionAfter (final int *a\_pos*) [protected]

Finds the first instruction line controller after the given point.

The line with the given number is included in the search.

**Parameters:**

*a\_pos* the position from which the search starts

**Returns:**

the line number of the first position that is an instruction line, or -1 in case there is no instruction line after the given point

References      [umbra.instructions.BytecodeControllerContainer.getLineController\(\)](#),  
[umbra.instructions.BytecodeControllerContainer.my\\_editor\\_lines](#), and [umbra.instructions.BytecodeControllerContainer.my\\_instructions](#).

Here is the call graph for this function:



#### 6.14.3.9 final void umbra.instructions.BytecodeControllerContainer.appendInstructions (final LinkedList *the\_instructions*) [protected]

Adds the given list of [instructions](#) at the end of the local instruction list.

**Parameters:**

*the\_instructions* the [instructions](#) to be added

References [umbra.instructions.BytecodeControllerContainer.my\\_instructions](#).

#### 6.14.3.10 final void umbra.instructions.BytecodeControllerContainer.removeInstructionsInRegion (final int *the\_first*, final int *the\_last*) [protected]

Removes from the representation of the [instructions](#) the [instructions](#) contained in the given region.

The bounds of the region are included in the removal operation. We assume that the *first* and *the\_last* are within the range of available line numbers.

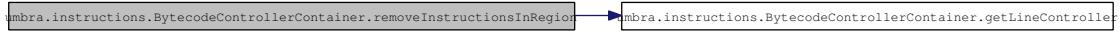
**Parameters:**

*the\_first* the first line of the region

*the\_last* the last line of the region

References umbra.instructions.BytecodeControllerContainer.getLineController(), and  
umbra.instructions.BytecodeControllerContainer.my\_instructions.

Here is the call graph for this function:



#### 6.14.3.11 final void umbra.instructions.BytecodeControllerContainer.insertInstructions (final int *a\_pos*, final LinkedList *the\_instructions*) [protected]

Insertst the given list of [instructions](#) at the given position.

The [instructions](#) after and including the postion *a\_pos* are shifted to the right (i.e. their indicies are increased) the number of positions that is enough to cover the given list *the\_instructions*.

**Parameters:**

*a\_pos* the postion where the list is inserted

*the\_instructions* the list to insert

References umbra.instructions.BytecodeControllerContainer.my\_instructions.

#### 6.14.3.12 final void umbra.instructions.BytecodeControllerContainer.insertEditorLine (final int *a\_pos*, final BytecodeLineController *a\_lc*) [protected]

Insertst the given line controller at the given position.

The instruction at the postion *a\_pos* and all [instructions](#) after that are shifted to the right (i.e. their indicies are incremented).

**Parameters:**

*a\_pos* the position at which the controller is inserted

*a\_lc* the controller to be inserted

References umbra.instructions.BytecodeControllerContainer.my\_editor\_lines.

#### 6.14.3.13 final int umbra.instructions.BytecodeControllerContainer.getAnnotationEnd (final int *a\_pos*) [protected]

Returns the last annotation line for the annotation lines block starting with the given position.

We assume the given position points to an [AnnotationLineController](#).

**Parameters:**

*a\_pos* a position with an annotation line controller

**Returns:**

the position of the last annotation line controller in the current block

References      `umbra.instructions.BytecodeControllerContainer.getLineController()`,      and  
`umbra.instructions.BytecodeControllerContainer.my_editor_lines`.

Referenced by `umbra.instructions.BytecodeController建立CurrentContext()`.

Here is the call graph for this function:



#### 6.14.3.14 final void umbra.instructions.BytecodeControllerContainer.controlPrint (final int *an\_index*) [protected]

This is a helper method used for debugging purposes.

It prints out all the [instructions](#) in the internal Umbra representation of a class file.

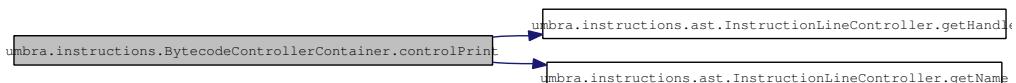
**Parameters:**

*an\_index* the number which allows to make different printouts

References      `umbra.instructions.ast.InstructionLineController.getHandle()`,      um-  
`bra.instructions.ast.InstructionLineController.getName()`, and `umbra.instructions.BytecodeControllerContainer.my_instructions`.

Referenced by `umbra.instructions.BytecodeControllerContainer.init()`.

Here is the call graph for this function:



## 6.14.4 Member Data Documentation

#### 6.14.4.1 LinkedList umbra.instructions.BytecodeControllerContainer.my\_editor\_lines [private]

The list of all the lines in the current byte code [editor](#).

These lines are stored as objects the classes of which are subclasses of [BytecodeLineController](#).

Referenced by [umbra.instructions.BytecodeControllerContainer.getAnnotationEnd\(\)](#),  
[umbra.instructions.BytecodeControllerContainer.getFirstInstructionAfter\(\)](#),  
[umbra.instructions.BytecodeControllerContainer.getLineController\(\)](#), um-  
[umbra.instructions.BytecodeControllerContainer.getLineControllerNo\(\)](#), um-  
[umbra.instructions.BytecodeControllerContainer.init\(\)](#), [umbra.instructions.BytecodeControllerContainer.insertEditorLine\(\)](#),  
and [umbra.instructions.BytecodeControllerContainer.replaceLineController\(\)](#).

#### 6.14.4.2 LinkedList [umbra.instructions.BytecodeControllerContainer.my\\_instructions](#)

[private]

The list of all the lines in the [editor](#) which contain codes of [instructions](#).

These are represented as objects the classes of which are subclasses of [InstructionLineController](#).

Referenced by [umbra.instructions.BytecodeControllerContainer.appendInstructions\(\)](#), um-  
[umbra.instructions.BytecodeControllerContainer.controlPrint\(\)](#), [umbra.instructions.BytecodeControllerContainer.getFirstInstruction\(\)](#), um-  
[umbra.instructions.BytecodeControllerContainer.getFirstInstructionInRegion\(\)](#), um-  
[umbra.instructions.BytecodeControllerContainer.getInstruction\(\)](#), [umbra.instructions.BytecodeControllerContainer.getNoOfInstructions\(\)](#),  
[umbra.instructions.BytecodeControllerContainer.init\(\)](#), [umbra.instructions.BytecodeControllerContainer.insertInstructions\(\)](#),  
and [umbra.instructions.BytecodeControllerContainer.removeInstructionsInRegion\(\)](#).

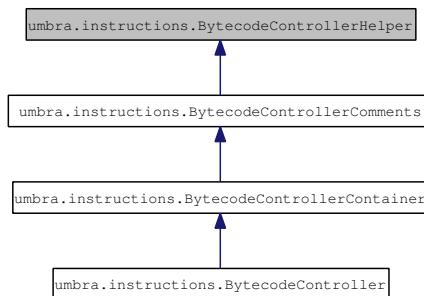
The documentation for this class was generated from the following file:

- source/umbra/instructions/[BytecodeControllerContainer.java](#)

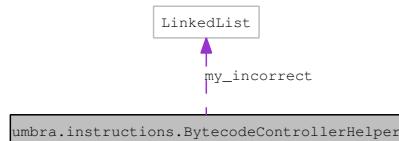
## 6.15 umbra.instructions.BytecodeControllerHelper Class Reference

This class contains various helper methods that are used in the [BytecodeController](#) class.

Inheritance diagram for `umbra.instructions.BytecodeControllerHelper`:



Collaboration diagram for `umbra.instructions.BytecodeControllerHelper`:



### Public Member Functions

- void [removeIncorrects](#) (final int a\_start, final int a\_stop)

*The method removes from the collection of the incorrect lines all the lines which are between a\_start and a\_stop.*

- boolean [allCorrect](#) ()
- boolean [bodyCorrect](#) ()

*Returns the information about the correctness of the method bodies in the current controller.*

- int [getFirstError](#) ()
- boolean[ ] [getModified](#) ()

*Returns the information on which methods were modified in the editor.*

- void [setModified](#) (final boolean[ ] the\_modified)
- void [initModTable](#) ()

*This method causes the initialisation of the table which keeps track of the modified methods.*

### Static Public Member Functions

- static void [showEditorLines](#) (final LinkedList the\_list)

*This is a debugging method.*

- static void [showAllIncorrectLines](#) (final LinkedList the\_list)

*This method prints out to the standard output the list of all the incorrect [instructions](#) in the controller.*

## Protected Member Functions

- [BytecodeControllerHelper](#) ()  
*This constructor initialises the internal container of the incorrect lines to be empty.*
- void [addIncorrect](#) (final BytecodeLineController a\_bcl)  
*Adds at the end of the incorrect lines list the given line controller.*
- abstract BytecodeLineController [getLineController](#) (final int a\_lineno)  
*Returns the line controller for the given line.*
- abstract int [getLineControllerNo](#) (final BytecodeLineController a\_line)  
*Returns the line number for the given line.*
- void [markModified](#) (final int a\_methno)  
*Marks given method as modified.*
- void [fillModTable](#) (final int a\_methodnum)  
*Fills in the structure which keeps track of the modified methods.*

## Private Attributes

- LinkedList [my\\_incorrect](#)  
*The list of all the lines which were detected to be incorrect.*
- boolean[ ] [my\\_modified](#)  
*Keeps track of modified methods.*

### 6.15.1 Detailed Description

This class contains various helper methods that are used in the [BytecodeController](#) class.

It also keeps track of the incorrect lines and the modified lines.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

## 6.15.2 Constructor & Destructor Documentation

### 6.15.2.1 umbra.instructions.BytecodeControllerHelper.BytecodeControllerHelper () [protected]

This constructor initialises the internal container of the incorrect lines to be empty.

The structures to keep track of the modified lines are left uninitialized.

References umbra.instructions.BytecodeControllerHelper.my\_incorrect.

## 6.15.3 Member Function Documentation

### 6.15.3.1 static void umbra.instructions.BytecodeControllerHelper.showEditorLines (final LinkedList *the\_list*) [static]

This is a debugging method.

It prints out to the standard output the all the controllers in the given list.

**Parameters:**

*the\_list* the list of line controllers

### 6.15.3.2 static void umbra.instructions.BytecodeControllerHelper.showAllIncorrectLines (final LinkedList *the\_list*) [static]

This method prints out to the standard output the list of all the incorrect [instructions](#) in the controller.

We assume the calls to this method are guarded by checks of [umbra.lib.FileName#DEBUG\\_MODE](#).

**Parameters:**

*the\_list* the list of controllers to present as incorrect ones

### 6.15.3.3 void umbra.instructions.BytecodeControllerHelper.removeIncorrects (final int *a\_start*, final int *a\_stop*)

The method removes from the collection of the incorrect lines all the lines which are between *a\_start* and *a\_stop*.

**Parameters:**

*a\_start* the first line which is checked for removing  
*a\_stop* the last line which is checked for removing

References umbra.instructions.BytecodeControllerHelper.getLineController(), and  
umbra.instructions.BytecodeControllerHelper.my\_incorrect.

Referenced by [umbra.editor.BytecodeDocument.updateFragment\(\)](#).

Here is the call graph for this function:



#### 6.15.3.4 boolean umbra.instructions.BytecodeControllerHelper.allCorrect ()

**Returns:**

`true` if there is no incorrect line within the whole document

References `umbra.instructions.BytecodeControllerHelper.my_incorrect`.

Referenced by `umbra.instructions.BytecodeControllerHelper.bodyCorrect()`.

#### 6.15.3.5 boolean umbra.instructions.BytecodeControllerHelper.bodyCorrect ()

Returns the information about the correctness of the method bodies in the current controller.

**Returns:**

`true` when the method bodies are syntactically correct and `false` otherwise

References `umbra.instructions.BytecodeControllerHelper.allCorrect()`.

Referenced by `umbra.editor.BytecodeContribution.BytecodeListener.documentChanged()`.

Here is the call graph for this function:



#### 6.15.3.6 int umbra.instructions.BytecodeControllerHelper.getFirstError ()

**Returns:**

number of a line that the first error occurs (not necessarily: number of the first line that an error occurs)

References `umbra.instructions.BytecodeControllerHelper.getLineControllerNo()`, and  
`umbra.instructions.BytecodeControllerHelper.my_incorrect`.

Referenced by `umbra.editor.BytecodeContribution.BytecodeListener.documentChanged()`.

Here is the call graph for this function:



#### 6.15.3.7 void umbra.instructions.BytecodeControllerHelper.addIncorrect (final BytecodeLineController *a\_bcl*) [protected]

Adds at the end of the incorrect lines list the given line controller.

**Parameters:**

*a\_bcl* the controller to add

References `umbra.instructions.BytecodeControllerHelper.my_incorrect`.

**6.15.3.8 abstract BytecodeLineController umbra.instructions.BytecodeControllerHelper.getLineController (final int a\_lineno) [protected, pure virtual]**

Returns the line controller for the given line.

**Parameters:**

*a\_lineno* the line number of the retrieved controller line

**Returns:**

the controller line for the given line number

Implemented in [umbra.instructions.BytecodeControllerContainer](#).

Referenced by [umbra.instructions.BytecodeControllerHelper.removeIncorrects\(\)](#).

**6.15.3.9 abstract int umbra.instructions.BytecodeControllerHelper.getLineControllerNo (final BytecodeLineController a\_line) [protected, pure virtual]**

Returns the line number for the given line.

**Parameters:**

*a\_line* the line controller for which we obtain the number of line

**Returns:**

the number of line for the given controller or -1 if there is no such a line

Implemented in [umbra.instructions.BytecodeControllerContainer](#).

Referenced by [umbra.instructions.BytecodeControllerHelper.getFirstError\(\)](#).

**6.15.3.10 void umbra.instructions.BytecodeControllerHelper.markModified (final int a\_methno) [protected]**

Marks given method as modified.

**Parameters:**

*a\_methno* the number of the marked method

References [umbra.instructions.BytecodeControllerHelper.my\\_modified](#).

**6.15.3.11 boolean [] umbra.instructions.BytecodeControllerHelper.getModified ()**

Returns the information on which methods were modified in the [editor](#).

This is used to enable the possibility to replace the code of the methods modified on the source code level, but that were not modified at the byte code level. See [umbra.editor.actions.BytecodeCombineAction](#). The returned array has `true` in entries that correspond to modified methods and `false` otherwise.

**Returns:**

the array with information on modified methods.

References `umbra.instructions.BytecodeControllerHelper.my_modified`.

Referenced by `umbra.editor.actions.BytecodeRefreshAction.doRefresh()`, `umbra.editor.actions.history.BytecodeRestoreAction.refreshContent()`, and `umbra.editor.actions.BytecodeCombineAction.updateModifiedMethods()`.

#### **6.15.3.12 void `umbra.instructions.BytecodeControllerHelper.setModified (final boolean[ ] the_modified)`**

**Parameters:**

*the\_modified* the array that indicates which methods were modified

Referenced by `umbra.editor.actions.BytecodeRefreshAction.doRefresh()`, and `umbra.editor.actions.history.BytecodeRestoreAction.refreshContent()`.

#### **6.15.3.13 void `umbra.instructions.BytecodeControllerHelper.initModTable ()`**

This method causes the initialisation of the table which keeps track of the modified methods.

References `umbra.instructions.BytecodeControllerHelper.my_modified`.

#### **6.15.3.14 void `umbra.instructions.BytecodeControllerHelper.fillModTable (final int a_methodnum) [protected]`**

Fills in the structure which keeps track of the modified methods.

**Parameters:**

*a\_methodnum* the number of the method to modify

References `umbra.instructions.BytecodeControllerHelper.my_modified`.

Referenced by `umbra.instructions.BytecodeControllerContainer.init()`.

### **6.15.4 Member Data Documentation**

#### **6.15.4.1 `LinkedList umbra.instructions.BytecodeControllerHelper.my_incorrect [private]`**

The list of all the lines which were detected to be incorrect.

Referenced by `umbra.instructions.BytecodeControllerHelper.addIncorrect()`, `umbra.instructions.BytecodeControllerHelper.allCorrect()`, `umbra.instructions.BytecodeControllerHelper[BytecodeControllerHelp]`, `umbra.instructions.BytecodeControllerHelper.getFirstError()`, and `umbra.instructions.BytecodeControllerHelper.removeInorrects()`.

#### **6.15.4.2 `boolean [ ] umbra.instructions.BytecodeControllerHelper.my_modified [private]`**

Keeps track of modified methods.

Each time a method is modified an entry with the method number is marked `true` in the array. The field is first initialised to be `null`. This field is initialised by a separate method - not within the constructor.

Referenced by `umbra.instructions.BytecodeControllerHelper.fillModTable()`, `umbra.instructions.BytecodeControllerHelper.getModified()`, `umbra.instructions.BytecodeControllerHelper.initModTable()`, and `umbra.instructions.BytecodeControllerHelper.markModified()`.

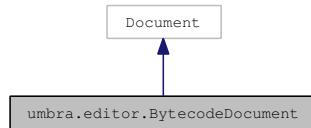
The documentation for this class was generated from the following file:

- source/umbra/instructions/[BytecodeControllerHelper.java](#)

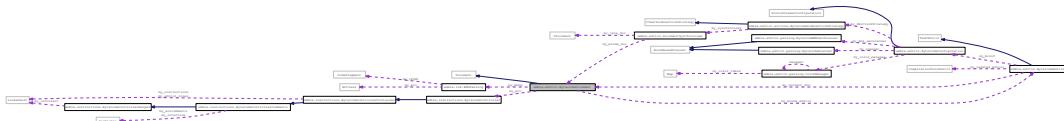
## 6.16 umbra.editor.BytocodeDocument Class Reference

This class is an abstract model of a byte code textual document.

Inheritance diagram for umbra.editor.BytocodeDocument:



Collaboration diagram for umbra.editor.BytocodeDocument:



### Public Member Functions

- [BytecodeDocument \(\)](#)

*This constructor creates a [BytecodeDocument](#) and associates a fresh, non-initialised model of the document.*

- final JavaClass [getJavaClass \(\)](#)
- final ClassGen [getClassGen \(\)](#)

*The method returns the [ClassGen](#) object for the current representation of the Java class file.*

- final void [setEditor \(final BytecodeEditor an\\_editor, final BMLParsing a\\_bmlp\)](#)

*This method updates the byte code [editor](#) associated with the current document.*

- final [BytecodeEditor getEditor \(\)](#)
- final boolean [isListenerAdded \(\)](#)
- boolean [isReady \(\)](#)

*Informs if the internal data structures that provide the model of the document are initialised.*

- void [init \(final String\[ \] a\\_comment\\_array, final String\[ \] an\\_interline\)](#) throws UmbraLocationException, UmbraMethodException

*This method initialises the internal structures of the byte code controller.*

- void [updateFragment \(final int a\\_start, final int an\\_oldend, final int a\\_newend\)](#) throws UmbraException, UmbraLocationException

*The method updates the internal structures of the document to reflect the change.*

- [BMLParsing getBmlp \(\)](#)
- void [updateJavaClass \(\)](#)

*Commits all the changes in the BMLLib representation of a class to a BCEL [JavaClass](#) object which is responsible for saving the class content to a class file.*

- `String printCode ()`  
*This method returns the textual representation of the byte code.*
- `MethodGen getMethodGen (final int a_method_no)` throws `UmbraMethodException`  
*Returns the `MethodGen` structure which handles the modifications in the method of the given number.*
- `boolean isInInit ()`  
*This method checks if the current document is in the initialisation process so that the changes of its content should not be processed.*
- `void setTextWithDeadUpdate (final String a_string)`  
*This method changes the content of this document in such a way that the update of the internal structures is not done.*
- `boolean annotCorrect ()`  
*Returns the information about the correctness of the last edited annotation in the current document.*
- `String getAnnotError ()`  
*Returns the error message for the last edited annotation in the current document.*
- `BytecodeController getModel ()`  
*Returns the abstract representation of the document contents.*

## Private Attributes

- `BytecodeEditor my_bcode_editor`  
*The byte code `editor` that manipulates the current document.*
- `BytecodeController my_bcc`  
*The object which contains the internal Umbra representation of the current document.*
- `boolean my_ready_flag`  
*This flag is true when the internal structures that connect the text .btc file with the BCEL representation are initialised.*
- `BMLParsing my_bmlp`  
*BML-annotated byte code (text + AST) displayed in this `editor`.*
- `boolean my_is_in_init`  
*It is true when the processing is inside the initialisation of the document.*

### 6.16.1 Detailed Description

This class is an abstract model of a byte code textual document.

It mainly handles the synchronisation between a byte code file and a Java source code file (in both directions).

It mediates between an `editor` which edits the document and the line structure of the byte code document. It also provides the connection with BMLLib structures.

**Author:**

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

**Version:**

a-01

## 6.16.2 Constructor & Destructor Documentation

### 6.16.2.1 `umbra.editor.BytecodeDocument.BytecodeDocument ()`

This constructor creates a `BytecodeDocument` and associates a fresh, non-initialised model of the document.

## 6.16.3 Member Function Documentation

### 6.16.3.1 `final JavaClass umbra.editor.BytecodeDocument.getJavaClass ()`

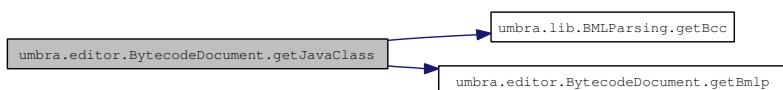
**Returns:**

the current representation of the Java class associated with the document.

References `umbra.lib.BMLParsing.getBcc()`, and `umbra.editor.BytecodeDocument.getBmlp()`.

Referenced by `umbra.editor.BytecodeEditor.doSave()`, `umbra.editor.DocumentSynchroniser.synchronizeBS()`, and `umbra.editor.actions.BytecodeCombineAction.updateMethodsLogic()`.

Here is the call graph for this function:



### 6.16.3.2 `final ClassGen umbra.editor.BytecodeDocument.getClassGen ()`

The method returns the `ClassGen` object for the current representation of the Java class file.

Each time this method is called a new object is generated.

**Returns:**

the current generator of the Java class file

References `umbra.editor.BytecodeDocument.getBmlp()`.

Here is the call graph for this function:



### 6.16.3.3 final void umbra.editor.BytecodeDocument.setEditor (final BytecodeEditor *an\_editor*, final BMLParsing *a\_bmlp*)

This method updates the byte code `editor` associated with the current document.

Additionally, it updates the fields that contain the representation of BML.

**Parameters:**

*an\_editor* the byte code `editor`

*a\_bmlp* a BMLLib representation of the current class

References `umbra.editor.BytecodeDocument.my_bcode_editor`, `umbra.editor.BytecodeDocument.my_bmlp`, and `umbra.editor.BytecodeEditor.setDocument()`.

Referenced by `umbra.editor.BytecodeEditorContributor.refreshEditor()`, and `umbra.editor.BytecodeDocumentProvider.setRelation()`.

Here is the call graph for this function:



### 6.16.3.4 final BytecodeEditor umbra.editor.BytecodeDocument.getEditor ()

**Returns:**

the `editor` for the current byte code document

References `umbra.editor.BytecodeDocument.my_bcode_editor`.

Referenced by `umbra.editor.actions.BytecodeDoubleClickStrategy.doubleClicked()`, `umbra.editor.parsing.NonRuleBasedDamagerRepairer.getDamageRegion()`, `umbra.editor.actions.BytecodeDoubleClickStrategy.getDocSync()`, and `umbra.editor.DocumentSynchroniser.synchronizeBS()`.

### 6.16.3.5 final boolean umbra.editor.BytecodeDocument.isListenerAdded ()

**Returns:**

`true` when the document change listener has already been added to the document

Referenced by `umbra.editor.BytecodeContribution.addListener()`.

### 6.16.3.6 boolean umbra.editor.BytecodeDocument.isReady ()

Informs if the internal data structures that provide the model of the document are initialised.

**Returns:**

`true` when the structures are initialised, `false` otherwise

References `umbra.editor.BytecodeDocument.my_ready_flag`.

Referenced by `umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged()`.

### 6.16.3.7 void umbra.editor.BytecodeDocument.init (final String[ ] *a\_comment\_array*, final String[ ] *an\_interline*) throws UmbraLocationException, UmbraMethodException

This method initialises the internal structures of the byte code controller.

In particular it initialises the object that manages the BCEL operations and enables the relevant [actions](#) in the Umbra plugin byte code contributor.

#### Parameters:

*a\_comment\_array* contains the texts of end-of-line comments, the i-th entry contains the comment for the i-th instruction in the file, if this parameter is null then the array is not taken into account  
*an\_interline* as above for multi-line comments

#### Exceptions:

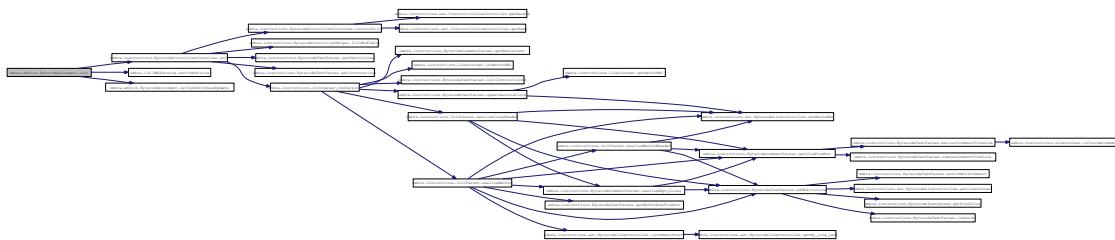
**UmbraLocationException** thrown in case a position has been reached which is outside the current document

**UmbraMethodException** in case the textual representation has more methods than the internal one

References `umbra.instructions.BytecodeControllerContainer.init()`, `umbra.editor.BytecodeDocument.my_bcc`, `umbra.editor.BytecodeDocument.my_bmlp`, `umbra.editor.BytecodeDocument.my_ready_flag`, `umbra.lib.BMLParsing.setCodeString()`, and `umbra.editor.BytecodeDocument.setTextWithDeadUpdate()`.

Referenced by `umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged()`, and `umbra.editor.BytecodeEditorContributor.refreshEditor()`.

Here is the call graph for this function:



### 6.16.3.8 void umbra.editor.BytecodeDocument.updateFragment (final int *a\_start*, final int *an\_oldend*, final int *a\_newend*) throws UmbraException, UmbraLocationException

The method updates the internal structures of the document to reflect the change.

The change is already present in the textual representation of the document.

#### Parameters:

*a\_start* the first changed line  
*an\_oldend* the last line of the change in the old version of the document  
*a\_newend* the last line of the change in the current version of the document

#### Exceptions:

**UmbraException** in case the change cannot be incorporated into the internal structures

***UmbraLocationException*** thrown in case a position has been reached which is outside the current document

References `umbra.editor.BytecodeDocument.my_bcc`, and `umbra.instructions.BytecodeControllerHelper.removeIncorrects()`. Referenced by `umbra.editor.BytecodeContribution.BytecodeListener.updateFragment()`.

Here is the call graph for this function:



### 6.16.3.9 BMLParsing `umbra.editor.BytecodeDocument.getBmlp()`

**Returns:**

BML-annotated byte code (text + AST) displayed in this [editor](#). All byte code modifications should be made through this object.

**See also:**

[BMLParsing](#)

References `umbra.editor.BytecodeDocument.my_bmlp`.

Referenced by `umbra.editor.BytecodeDocument.getClassGen()`, `umbra.editor.BytecodeDocument.getJavaClass()`, `umbra.editor.BytecodeDocument.printCode()`, `umbra.editor.BytecodeEditor.setRelation()`, and `umbra.editor.BytecodeDocument.updateJavaClass()`.

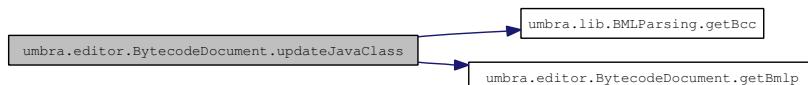
### 6.16.3.10 void `umbra.editor.BytecodeDocument.updateJavaClass()`

Commits all the changes in the BMLLib representation of a class to a BCEL [JavaClass](#) object which is responsible for saving the class content to a class file.

References `umbra.lib.BMLParsing.getBcc()`, and `umbra.editor.BytecodeDocument.getBmlp()`.

Referenced by `umbra.editor.BytecodeEditor.doSave()`.

Here is the call graph for this function:



### 6.16.3.11 String `umbra.editor.BytecodeDocument.printCode()`

This method returns the textual representation of the byte code.

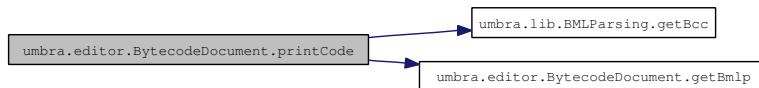
The textual representation is generated from the BMLlib structures.

**Returns:**

the textual representation of the byte code

References `umbra.lib.BMLParsing.getBcc()`, and `umbra.editor.BytecodeDocument.getBmlp()`.

Here is the call graph for this function:



#### **6.16.3.12 MethodGen `umbra.editor.BytecodeDocument.getMethodGen (final int a_method_no)` throws `UmbraMethodException`**

Returns the [MethodGen](#) structure which handles the modifications in the method of the given number.

**Parameters:**

`a_method_no` the number of the method to be returned

**Returns:**

the BCEL structure which handles the editing of the given method

**Exceptions:**

*UmbraMethodException* thrown in case the given method number is outside the range of available methods

References `umbra.lib.BMLParsing.getBcc()`, and `umbra.editor.BytecodeDocument.my_bmlp`.

Here is the call graph for this function:



#### **6.16.3.13 boolean `umbra.editor.BytecodeDocument.isInInit ()`**

This method checks if the current document is in the initialisation process so that the changes of its content should not be processed.

**Returns:**

`true` when the document is in the initialisation process, `false` otherwise

References `umbra.editor.BytecodeDocument.my_is_in_init`.

Referenced by `umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged()`, and `umbra.editor.BytecodeContribution.BytecodeListener.documentChanged()`.

#### **6.16.3.14 void `umbra.editor.BytecodeDocument.setTextWithDeadUpdate (final String a_string)`**

This method changes the content of this document in such a way that the update of the internal structures is not done.

This is used when the initial structure is generated.

**Parameters:**

*a\_string* the text of the document

References umbra.editor.BytecodeDocument.my\_is\_in\_init.

Referenced by umbra.editor.BytecodeDocument.init().

**6.16.3.15 boolean umbra.editor.BytecodeDocument.annotCorrect ()**

Returns the information about the correctness of the last edited annotation in the current document.

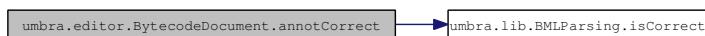
**Returns:**

`true` when the last annotation is syntactically correct and `false` otherwise

References umbra.lib.BMLParsing.isCorrect(), and umbra.editor.BytecodeDocument.my\_bmlp.

Referenced by umbra.editor.BytecodeContribution.BytecodeListener.documentChanged().

Here is the call graph for this function:

**6.16.3.16 String umbra.editor.BytecodeDocument.getAnnotError ()**

Returns the error message for the last edited annotation in the current document.

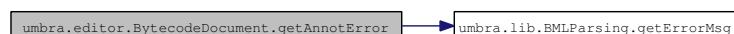
**Returns:**

`true` when the last annotation is syntactically correct and `false` otherwise

References umbra.lib.BMLParsing.getErrorMsg(), and umbra.editor.BytecodeDocument.my\_bmlp.

Referenced by umbra.editor.BytecodeContribution.BytecodeListener.documentChanged().

Here is the call graph for this function:

**6.16.3.17 BytecodeController umbra.editor.BytecodeDocument.getModel ()**

Returns the abstract representation of the document contents.

**Returns:**

the abstract representation of the document contents

References umbra.editor.BytecodeDocument.my\_bcc.

Referenced by umbra.editor.BytecodeContribution.BytecodeListener.documentChanged(), umbra.editor.actions.BytecodeRefreshAction.doRefresh(), umbra.java.actions.DisasBCEL.openBCodeEditorForJavaFile(), umbra.editor.actions.history.BytecodeRestoreAction.refreshContent(), umbra.editor.DocumentSynchroniser.syncBS(), and umbra.editor.actions.BytecodeCombineAction.updateModifiedMethods().

## 6.16.4 Member Data Documentation

### 6.16.4.1 BytecodeEditor **umbra.editor.BytecodeDocument.my\_bcode\_editor** [private]

The byte code [editor](#) that manipulates the current document.

Referenced by [umbra.editor.BytecodeDocument.getEditor\(\)](#), and [umbra.editor.BytecodeDocument.setEditor\(\)](#).

### 6.16.4.2 BytecodeController **umbra.editor.BytecodeDocument.my\_bcc** [private]

The object which contains the internal Umbra representation of the current document.

Referenced by [umbra.editor.BytecodeDocument.getModel\(\)](#), [umbra.editor.BytecodeDocument.init\(\)](#), and [umbra.editor.BytecodeDocument.updateFragment\(\)](#).

### 6.16.4.3 boolean **umbra.editor.BytecodeDocument.my\_ready\_flag** [private]

This flag is `true` when the internal structures that connect the text .btc file with the BCEL representation are initialised.

Referenced by [umbra.editor.BytecodeDocument.init\(\)](#), and [umbra.editor.BytecodeDocument.isReady\(\)](#).

### 6.16.4.4 BMLParsing **umbra.editor.BytecodeDocument.my\_bmlp** [private]

BML-annotated byte code (text + AST) displayed in this [editor](#).

All byte code modifications should be made on this object.

Referenced by [umbra.editor.BytecodeDocument.annotCorrect\(\)](#), [umbra.editor.BytecodeDocument.getAnnotError\(\)](#), [umbra.editor.BytecodeDocument.getBmlp\(\)](#), [umbra.editor.BytecodeDocument.getMethodGen\(\)](#), [umbra.editor.BytecodeDocument.init\(\)](#), and [umbra.editor.BytecodeDocument.setEditor\(\)](#).

### 6.16.4.5 boolean **umbra.editor.BytecodeDocument.my\_is\_in\_init** [private]

It is true when the processing is inside the initialisation of the document.

This is to forbid double initialisation inside the init method.

Referenced by [umbra.editor.BytecodeDocument.isInInit\(\)](#), and [umbra.editor.BytecodeDocument.setTextWithDeadUpdate\(\)](#).

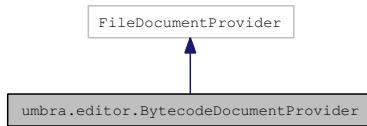
The documentation for this class was generated from the following file:

- [source/umbra/editor/BytecodeDocument.java](#)

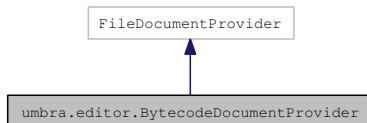
## 6.17 umbra.editor.BytecodeDocumentProvider Class Reference

This class has been modified with relation to the generated automatically to allow adding listener that is responsible for error checking.

Inheritance diagram for umbra.editor.BytecodeDocumentProvider:



Collaboration diagram for umbra.editor.BytecodeDocumentProvider:



### Public Member Functions

- final void [setRelation](#) (final CompilationUnitEditor an\_editor, final [BytecodeEditor](#) a\_bcode\_editor, final IEeditorInput an\_input, final [BMLParsing](#) a\_bmlp)

*This method creates connection between the document specified by an\_input object and given editors.*

### Protected Member Functions

- final IDocument [createEmptyDocument](#) ()

*This method creates a byte code document with the empty content.*

- final IDocument [createDocument](#) (final Object an\_element) throws CoreException

*The method used to create [IDocument](#) structure when the [editor](#) is initialised.*

### 6.17.1 Detailed Description

This class has been modified with relation to the generated automatically to allow adding listener that is responsible for error checking.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

## 6.17.2 Member Function Documentation

### 6.17.2.1 final IDocument umbra.editor.BytecodeDocumentProvider.createEmptyDocument () [protected]

This method creates a byte code document with the empty content.

#### Returns:

a fresh [BytecodeDocument](#) object with no content

Referenced by `umbra.editor.BytecodeDocumentProvider.createDocument()`.

### 6.17.2.2 final IDocument umbra.editor.BytecodeDocumentProvider.createDocument (final Object an\_element) throws CoreException [protected]

The method used to create [IDocument](#) structure when the [editor](#) is initialised.

This method checks if the parameter `an_element` has the type [IEditorInput](#). In case the type is proper it creates an empty document and then fills its contents with the data in the file associated with `an_element`. In case the file does not exists, an empty file is created first. Subsequently, the colouring of the document structure is set using [BytecodePartitionScanner](#). At last the document is added to the event listener associated with the byte code [editor](#) (i.e. the one in [BytecodeContribution](#)).

#### Parameters:

`an_element` an element for which we create the document, the actual type of this object should be [IEditorInput](#)

#### Returns:

the document structure or `null` in case the parameter `an_element` is null or is not [IEditorInput](#)

#### Exceptions:

[CoreException](#) if the input for `an_element` cannot be accessed or for the reasons presented in `boolean, org.eclipse.core.runtime.IProgressMonitor`

[org.eclipse.core.runtime.OperationCanceledException](#) in case the operation to create the new file was canceled, this may also happen in case no user canceled the operation

References [umbra.editor.BytecodeContribution.addListener\(\)](#), and [umbra.editor.BytecodeDocumentProvider.createEmptyDocument\(\)](#).

Here is the call graph for this function:



### 6.17.2.3 final void umbra.editor.BytecodeDocumentProvider.setRelation (final CompilationUnitEditor an\_editor, final BytecodeEditor a\_bcode\_editor, final IEditorInput an\_input, final BMLParsing a\_bmlp)

This method creates connection between the document specified by `an_input` object and given editors.

This method sets `a_bcode_editor` as the `editor` and `an_editor` as the related `editor` for a byte code document that works on `an_input`. Additionally, it adds the document to the event listener for the byte code `editor actions`.

**Parameters:**

- `an_editor` the `editor` of the Java source code
- `a_bcode_editor` the byte code `editor` in which the textual representation is to be edited
- `an_input` input file with the textual representation of the byte code
- `a_bmlp` a BMLLib representation of the class in the document

References `umbra.editor.BytecodeDocument.setEditor()`.

Here is the call graph for this function:



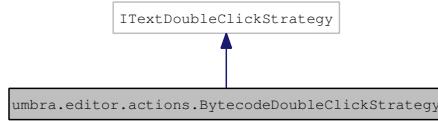
The documentation for this class was generated from the following file:

- `source/umbra/editor/BytecodeDocumentProvider.java`

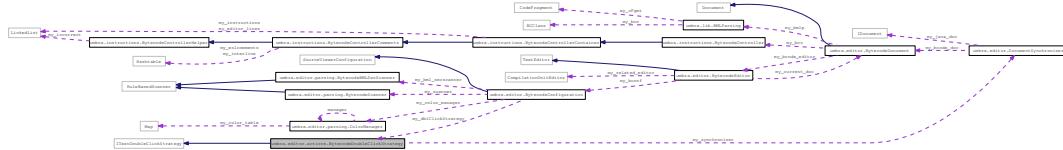
## 6.18 umbra.editor.actions.BytecodeDoubleClickStrategy Class Reference

This class is responsible for action that is performed after a double click event in a byte code [editor](#) window.

Inheritance diagram for `umbra.editor.actions.BytecodeDoubleClickStrategy`:



Collaboration diagram for `umbra.editor.actions.BytecodeDoubleClickStrategy`:



### Public Member Functions

- final void `doubleClicked` (final `ITextViewer` a\_selection)

*This method implements the reaction on the double click in a byte code [editor](#).*

### Private Member Functions

- `DocumentSynchroniser getDocSynch` (final `BytecodeDocument` a\_doc)

*This method lazily provides the object which performs the synchronisation operations.*

### Private Attributes

- `DocumentSynchroniser my_synchroniser`

*This is an object which handles the calculations of the synchronisation positions.*

### 6.18.1 Detailed Description

This class is responsible for action that is performed after a double click event in a byte code [editor](#) window. This triggers a synchronisation action which relates the position clicked within the byte code [editor](#) to the source code in the corresponding Java file [editor](#).

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

**See also:**[BytecodeDocument](#)**6.18.2 Member Function Documentation****6.18.2.1 final void umbra.editor.actions.BytecodeDoubleClickStrategy.doubleClicked (final ITextViewer *a\_selection*)**

This method implements the reaction on the double click in a byte code editor.

It synchronises the position clicked within the byte code editor to the source code in the corresponding Java file editor. Most the information about the selected area is not used. Only the position of the cursor is taken into account.

**Parameters:**

*a\_selection* the selected area of the byte code document

References umbra.editor.actions.BytecodeDoubleClickStrategy.getDocSynch(), umbra.editor.BytecodeDocument.getEditor(), and umbra.editor.DocumentSynchroniser.synchronizeBS().

Here is the call graph for this function:

**6.18.2.2 DocumentSynchroniser umbra.editor.actions.BytecodeDoubleClickStrategy.getDocSynch (final BytecodeDocument *a\_doc*) [private]**

This method lazily provides the object which performs the synchronisation operations.

**Parameters:**

*a\_doc* a byte code document for which the synchronisation is performed

**Returns:**

a [DocumentSynchroniser](#) which performs the synchronisation operations

References umbra.editor.BytecodeDocument.getEditor(), umbra.editor.BytecodeEditor.getRelatedEditor(), and umbra.editor.actions.BytecodeDoubleClickStrategy.my\_synchroniser.

Referenced by umbra.editor.actions.BytecodeDoubleClickStrategy.doubleClicked().

Here is the call graph for this function:



### 6.18.3 Member Data Documentation

#### 6.18.3.1 DocumentSynchroniser `umbra.editor.actions.BytecodeDoubleClickStrategy.my_-synchroniser` [private]

This is an object which handles the calculations of the synchronisation positions.

Referenced by `umbra.editor.actions.BytecodeDoubleClickStrategy.getDocSynch()`.

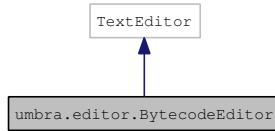
The documentation for this class was generated from the following file:

- source/umbra/editor/actions/[BytecodeDoubleClickStrategy.java](#)

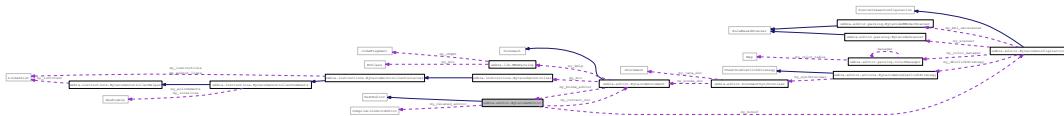
## 6.19 umbra.editor.BytecodeEditor Class Reference

This is the main class that defines the byte code `editor`.

Inheritance diagram for `umbra.editor.BytecodeEditor`:



Collaboration diagram for `umbra.editor.BytecodeEditor`:



### Public Member Functions

- `BytecodeEditor ()`

*This constructor creates the class and initialises the default colour manager.*

- `final void dispose ()`

*Default function used while closing the current `editor`.*

- `final CompilationUnitEditor getRelatedEditor ()`

*Returns the Java source code `editor` associated with the current byte code `editor`.*

- `final void setRelation (final CompilationUnitEditor an_editor)`

*This is a function executed directly after the initialisation and it arranges the relation between the `editor` and its source code counterpart.*

- `final void doSave (final IProgressMonitor a_progress_monitor)`

*This method is run automatically while standard Eclipse 'save' action is executed.*

- `final void refreshBytecode (final IPath a_path, final BytecodeDocument a_doc, final String[] the_comments, final String[] the_interline_comments) throws ClassNotFoundException, CoreException`

*This method loads in the content of the class file corresponding to the given Java source code file.*

- `final int newHistory ()`

*Updating number of historical versions executed after adding new version.*

- `final void clearHistory ()`

*Updating number of historical versions when all of them are removed.*

- `final void setDocument (final BytecodeDocument a_doc)`

- `final BytecodeDocument getDocument ()`

- void `setRelatedEditor` (final CompilationUnitEditor a\_related\_editor)
- void `renewConfiguration` (final BytecodeDocument a\_doc)

*This method creates new colouring configuration and associates this with the current editor.*

- int `getVisibleRegion` ()

*This method returns the number of the first visible line in the current textual byte code document.*

- void `setVisibleRegion` (final int a\_firstvisible)

*The method moves the content of the current textual byte code document so that the first visible line is the one given in the argument.*

## Protected Member Functions

- void `finalize` () throws Throwable

*This method disposes the colour allocated from the system and then calls the superclass finalisation.*

## Private Member Functions

- IFile `makeSpareCopy` ()

*This method saves the the current class file under a special name.*

- JavaClass `loadJavaClass` (final IPath a\_path, final SyntheticRepository a\_repo)

*This method loads from the given Java class repository a class pointed out by the given path.*

- SyntheticRepository `getCurrentClassRepository` () throws JavaModelException

*The method gives the repository where all the class files associated with the current project are located.*

## Private Attributes

- CompilationUnitEditor `my_related_editor`

*The Java source code editor that corresponds to the current byte code editor.*

- int `my_history_num` = -1

*This field contains the number of history items.*

- BytecodeConfiguration `my_bconf`

*The byte code editor configuration manager associated with the current editor.*

- BytecodeDocument `my_current_doc`

*Byte code document currently edited by the editor.*

### 6.19.1 Detailed Description

This is the main class that defines the byte code [editor](#).

It does so by subclassing [org.eclipse.ui.editors.text.TextEditor](#), which is a standard class extended by each [editor](#) plugin. Its additional features are attributes that describe BCEL structures related to the edited byte code such as [org.apache.bcel.classfile.JavaClass](#), to obtain particular [instructions](#), and [org.apache.bcel.generic.ClassGen](#), to allow changes in BCEL.

The input file for this [editor](#) is a .btc ([FileNames#BYTECODE\\_EXTENSION](#)) file which resides alongside the corresponding .java ([FileNames#JAVA\\_EXTENSION](#)) file. (Note that it is a different place from the place for .class, [FileNames#CLASS\\_EXTENSION](#), files).

#### Author:

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Wojciech Was ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

### 6.19.2 Constructor & Destructor Documentation

#### 6.19.2.1 [umbra.editor.BytecodeEditor.BytecodeEditor \(\)](#)

This constructor creates the class and initialises the default colour manager.

References [umbra.editor.BytecodeEditor.my\\_bconf](#).

### 6.19.3 Member Function Documentation

#### 6.19.3.1 [final void umbra.editor.BytecodeEditor.dispose \(\)](#)

Default function used while closing the current [editor](#).

#### 6.19.3.2 [final CompilationUnitEditor umbra.editor.BytecodeEditor.getRelatedEditor \(\)](#)

Returns the Java source code [editor](#) associated with the current byte code [editor](#).

#### Returns:

the Java source code [editor](#) that byte code text has been generated from

References [umbra.editor.BytecodeEditor.my\\_related\\_editor](#).

Referenced by [umbra.editor.actions.BytecodeSynchrAction.getDocSynch\(\)](#),  
[umbra.editor.actions.BytecodeDoubleClickStrategy.getDocSynch\(\)](#),  
[umbra.editor.actions.history.ClearHistoryAction.run\(\)](#), [umbra.editor.actions.history.BytecodeRestoreAction.run\(\)](#),  
and [umbra.editor.DocumentSynchroniser.synchronizeBS\(\)](#).

### 6.19.3.3 final void umbra.editor.BytecodeEditor.setRelation (final CompilationUnitEditor *an\_editor*)

This is a function executed directly after the initialisation and it arranges the relation between the [editor](#) and its source code counterpart.

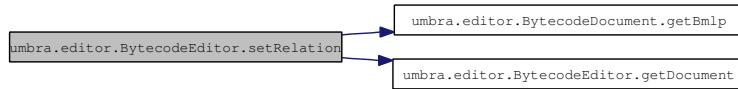
**Parameters:**

*an\_editor* Java code [editor](#) with intended relation (used in particular during synchronisation)

References [umbra.editor.BytecodeDocument.getBmlp\(\)](#), [umbra.editor.BytecodeEditor.getDocument\(\)](#), and [umbra.editor.BytecodeEditor.my\\_related\\_editor](#).

Referenced by [umbra.java.actions.DisasBCEL.openEditorAndDisassemble\(\)](#).

Here is the call graph for this function:



### 6.19.3.4 final void umbra.editor.BytecodeEditor.doSave (final IProgressMonitor *a\_progress\_monitor*)

This method is run automatically while standard Eclipse 'save' action is executed.

Additionally, the current class file is saved under the name with '\_' at the beginning for the later use (see [umbra.editor.actions.BytecodeRebuildAction](#) and [umbra.editor.actions.BytecodeCombineAction](#)). Except for that, the method updates structure [org.apache.bcel.classfile.JavaClass](#) in BCEL and binary files to make visible in the class file the changes made in the [editor](#).

**Parameters:**

*a\_progress\_monitor* not used

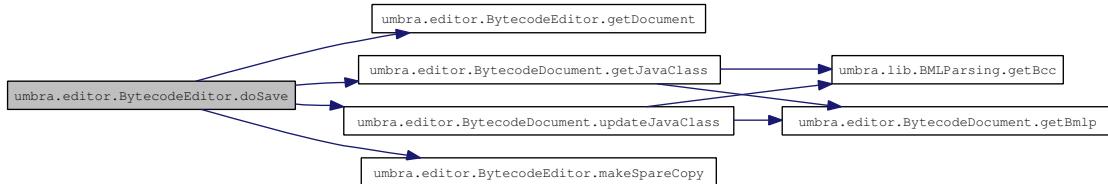
**See also:**

[org.eclipse.ui.texteditor.AbstractTextEditor.doSave\(IProgressMonitor\)](#)

References [umbra.editor.BytecodeEditor.getDocument\(\)](#), [umbra.editor.BytecodeDocument.getJavaClass\(\)](#), [umbra.editor.BytecodeEditor.makeSpareCopy\(\)](#), and [umbra.editor.BytecodeDocument.updateJavaClass\(\)](#).

Referenced by [umbra.editor.actions.BytecodeRefreshAction.run\(\)](#).

Here is the call graph for this function:



**6.19.3.5 IFile umbra.editor.BytecodeEditor.makeSpareCopy () [private]**

This method saves the the current class file under a special name.

This name consists of '\_' followed by the original name. The files of this kind are used in [umbra.editor.actions.BytecodeRebuildAction](#) and [umbra.editor.actions.BytecodeCombineAction](#).

**Returns:**

the IFile which points to the class file being edited by the current editor

Referenced by [umbra.editor.BytecodeEditor.doSave\(\)](#).

**6.19.3.6 final void umbra.editor.BytecodeEditor.refreshBytecode (final IPath *a\_path*, final BytecodeDocument *a\_doc*, final String[ ] *the\_comments*, final String[ ] *the\_interline\_comments*) throws ClassNotFoundException, CoreException**

This method loads in the content of the class file corresponding to the given Java source code file.

The method finds the class file corresponding to the given Java source code file, loads it to BCEL and BMLlib structures then it generates the .btc file with the textual representation of the class file. The BCEL and BMLlib representation of the class file is associated with the given document. Additionally, the comment information from the previous session is connected to the document.

**Parameters:**

*a\_path* a workspace relative path to a class file

*a\_doc* the byte code document for which the refresh operation is taken

*the\_comments* a table of end-of-line comments to be inserted

*the\_interline\_comments* table of comments between [instructions](#) to be also inserted

**Exceptions:**

**ClassNotFoundException** the class corresponding to the given path cannot be found

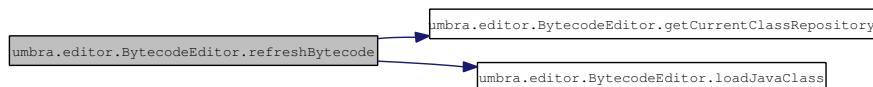
**CoreException** the reasons for this exception include:

- The location corresponding to the edited input in the local file system is occupied by a directory.
- The workspace is not in sync with the location of the input in the local file system and `force` is `false`.
- Resource changes are disallowed during certain types of resource change event notification. See [IResourceChangeEvent](#) for more details.
- The file modification validator of the input disallowed the change.
- The parent of this resource does not exist.
- The project of this resource is not accessible.
- The parent contains a resource of a different type at the same path as this resource.
- The name of this resource is not valid (according to [IWorkspace.validateName](#)).

References      [umbra.editor.BytecodeEditor.getCurrentClassRepository\(\)](#),      and      [umbra.editor.BytecodeEditor.loadJavaClass\(\)](#).

Referenced by [umbra.java.actions.DisasBCEL.openBCodeEditorForJavaFile\(\)](#),      and      [umbra.editor.actions.BytecodeCombineAction.updateMethodsLogic\(\)](#).

Here is the call graph for this function:



#### **6.19.3.7 JavaClass umbra.editor.BytecodeEditor.loadJavaClass (final IPath *a\_path*, final SyntheticRepository *a\_repo*) [private]**

This method loads from the given Java class repository a class pointed out by the given path.

**Parameters:**

- a\_path* a workspace relative path to the class file
- a\_repo* the repository to load the class from

**Returns:**

the BCEL [org.apache.bcel.classfile.JavaClass](#) structure with the content of the class file

Referenced by `umbra.editor.BytecodeEditor.refreshBytecode()`.

#### **6.19.3.8 SyntheticRepository umbra.editor.BytecodeEditor.getCurrentClassRepository () throws JavaModelException [private]**

The method gives the repository where all the class files associated with the current project are located.

**Returns:**

the repository of the class files

**Exceptions:**

*JavaModelException* if the output location for the current project does not exist

References `umbra.editor.BytecodeEditor.my_related_editor`.

Referenced by `umbra.editor.BytecodeEditor.refreshBytecode()`.

#### **6.19.3.9 final int umbra.editor.BytecodeEditor.newHistory ()**

Updating number of historical versions executed after adding new version.

**Returns:**

current number of versions; -1 if limit has been reached

References `umbra.editor.BytecodeEditor.my_history_num`.

**6.19.3.10 final void umbra.editor.BytecodeEditor.clearHistory ()**

Updating number of historical versions when all of them are removed.

References umbra.editor.BytecodeEditor.my\_history\_num.

Referenced by umbra.editor.actions.history.ClearHistoryAction.run().

**6.19.3.11 final void umbra.editor.BytecodeEditor.setDocument (final BytecodeDocument *a\_doc*)**

**Parameters:**

*a\_doc* document to associate with the current editor

References umbra.editor.BytecodeEditor.my\_current\_doc.

Referenced by umbra.editor.BytecodeDocument.setEditor().

**6.19.3.12 final BytecodeDocument umbra.editor.BytecodeEditor.getDocument ()**

**Returns:**

the currently edited document

References umbra.editor.BytecodeEditor.my\_current\_doc.

Referenced by umbra.editor.actions.BytecodeRefreshAction.doRefresh(), umbra.editor.BytecodeEditor.doSave(), umbra.editor.actions.history.BytecodeRestoreAction.refreshContent(), umbra.editor.actions.BytecodeRebuildAction.run(), umbra.editor.BytecodeEditor.setRelation(), umbra.editor.actions.BytecodeCombineAction.updateMethodsLogic(), and umbra.editor.actions.BytecodeCombineAction.updateModifiedMethods().

**6.19.3.13 void umbra.editor.BytecodeEditor.setRelatedEditor (final CompilationUnitEditor *a\_related\_editor*)**

**Parameters:**

*a\_related\_editor* the Java source code editor to associate with the current byte code editor

Referenced by umbra.java.actions.DisasBCEL.openBCodeEditorForJavaFile().

**6.19.3.14 void umbra.editor.BytecodeEditor.finalize () throws Throwable [protected]**

This method disposes the colour allocated from the system and then calls the superclass finalisation.

**Exceptions:**

*Throwable* in case something wrong happens in the superclass finalisation

**6.19.3.15 void umbra.editor.BytecodeEditor.renewConfiguration (final BytecodeDocument *a\_doc*)**

This method creates new colouring configuration and associates this with the current editor.

A new document is always created with default gray colouring mode. In case, we want to make use of the code colouring functionality, we must change that mode into another one. This is done with the help of this method which replaces the colouring logic with a one which is created here.

**Parameters:**

*a\_doc* the document for which we change the colouring

References `umbra.editor.BytecodeEditor.my_bconf`.

Referenced by `umbra.java.actions.DisasBCEL.openEditorAndDisassemble()`.

#### 6.19.3.16 int `umbra.editor.BytecodeEditor.getVisibleRegion ()`

This method returns the number of the first visible line in the current textual byte code document.

**Returns:**

the number of the first visible line

Referenced by `umbra.editor.actions.BytecodeRefreshAction.run()`.

#### 6.19.3.17 void `umbra.editor.BytecodeEditor.setVisibleRegion (final int a_firstvisible)`

The method moves the content of the current textual byte code document so that the first visible line is the one given in the argument.

**Parameters:**

*a\_firstvisible* the first line to be visible

Referenced by `umbra.editor.actions.BytecodeRefreshAction.run()`.

### 6.19.4 Member Data Documentation

#### 6.19.4.1 CompilationUnitEditor `umbra.editor.BytecodeEditor.my_related_editor` [private]

The Java source code `editor` that corresponds to the current byte code `editor`.

Referenced by `umbra.editor.BytecodeEditor.getCurrentClassRepository()`, `umbra.editor.BytecodeEditor.getRelatedEditor()`, and `umbra.editor.BytecodeEditor.setRelation()`.

#### 6.19.4.2 int `umbra.editor.BytecodeEditor.my_history_num = -1` [private]

This field contains the number of history items.

This field contains -1 when there are no active history snapshots (i.e. the history is clear).

Referenced by `umbra.editor.BytecodeEditor.clearHistory()`, and `umbra.editor.BytecodeEditor.newHistory()`.

**6.19.4.3 BytecodeConfiguration umbra.editor.BytecodeEditor.my\_bconf [private]**

The byte code [editor](#) configuration manager associated with the current [editor](#).

Referenced by umbra.editor.BytecodeEditor.BytecodeEditor(), and umbra.editor.BytecodeEditor.renewConfiguration().

**6.19.4.4 BytecodeDocument umbra.editor.BytecodeEditor.my\_current\_doc [private]**

Byte code document currently edited by the [editor](#).

Referenced by umbra.editor.BytecodeEditor.getDocument(), and umbra.editor.BytecodeEditor.setDocument().

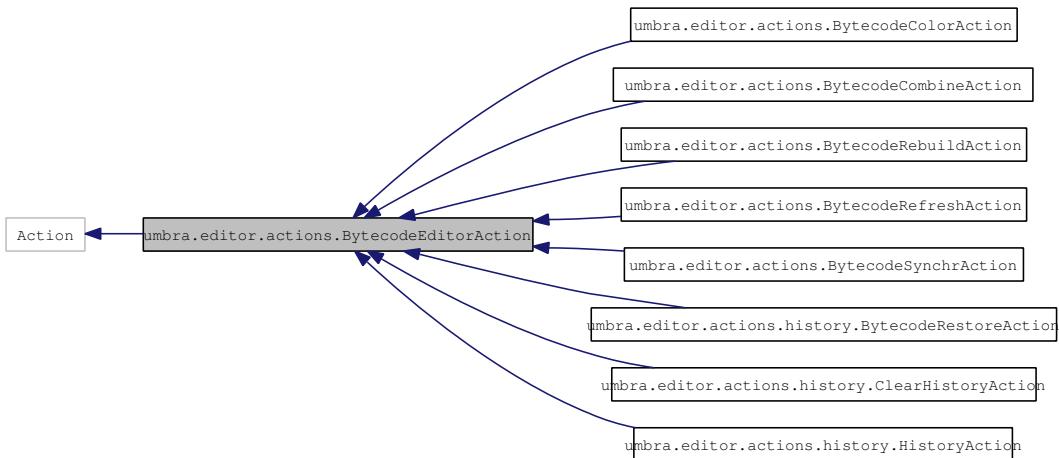
The documentation for this class was generated from the following file:

- source/umbra/editor/[BytecodeEditor.java](#)

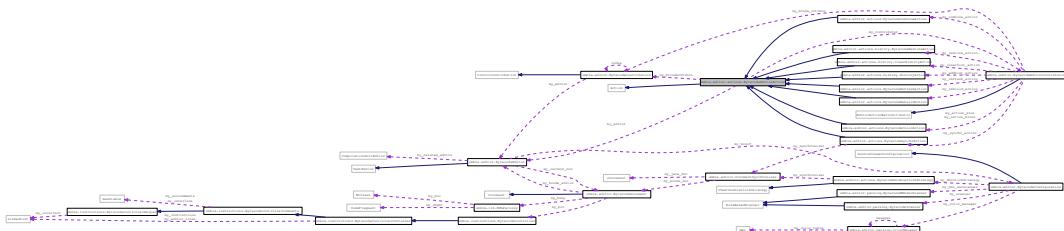
## 6.20 umbra.editor.actions.BytecodeEditorAction Class Reference

This class defines the common operations for all the byte code [editor actions](#).

Inheritance diagram for `umbra.editor.actions.BytecodeEditorAction`:



Collaboration diagram for `umbra.editor.actions.BytecodeEditorAction`:



### Public Member Functions

- `BytecodeEditorAction (final String a_name, final BytecodeEditorContributor a_contributor, final BytecodeContribution a_bytecode_contribution)`

*This constructor creates the generic part of a byte code [editor](#) action.*

- `void setActiveEditor (final IEeditorPart a_part)`

*The method sets the bytecode [editor](#) for which the operation will be performed.*

- `final BytecodeEditor getEditor ()`
- `final BytecodeEditorContributor getContributor ()`
- `final BytecodeContribution getContribution ()`

### Static Public Member Functions

- `static void wrongLocationMessage (final Shell a_shell, final String a_title, final UmbraLocationException an_ex)`

*Displays the message that a wrong location has been reached.*

- static void [wrongFileOperationMessage](#) (final Shell a\_shell, final String a\_title)  
*Displays the message that a file operation on a class file failed.*
- static void [wrongPathToClassMessage](#) (final Shell a\_shell, final String a\_title, final String a\_path)  
*Displays the message that a given path does not lead to a valid class file.*

## Private Attributes

- [BytecodeEditor my\\_editor](#)  
*The current byte code editor for which the action takes place.*
- [BytecodeEditorContributor my\\_contributor](#)  
*The manager that initialises all the actions within the byte code plugin.*
- [BytecodeContribution my\\_btcodeCntrbtn](#)  
*The GUI elements contributed to the eclipse GUI by the bytecode editor.*

### 6.20.1 Detailed Description

This class defines the common operations for all the byte code [editor actions](#).

It is responsible for accessing the [editor](#), contributor, and contribution. Except for that it contains the methods to display messages when errors occur.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.20.2 Constructor & Destructor Documentation

#### 6.20.2.1 umbra.editor.actions.BytecodeEditorAction.BytecodeEditorAction (final String a\_name, final BytecodeEditorContributor a\_contributor, final BytecodeContribution a\_bytecode\_contribution)

This constructor creates the generic part of a byte code [editor](#) action.

It registers the action under the title given by a\_name parameter and stores locally the object which creates all the [actions](#) and which contributes the [editor](#) GUI elements to the eclipse GUI.

#### Parameters:

*a\_name* a name of the action to register

*a\_contributor* the manager that initialises all the [actions](#) within the byte code plugin

*a\_bytecode\_contribution* the GUI elements contributed to the eclipse GUI by the byte code [editor](#).  
This reference should be the same as in the parameter *a\_contributor*.

References        [umbra.editor.actions.BytecodeEditorAction.my\\_btcodeCnrbtn](#),        and  
[umbra.editor.actions.BytecodeEditorAction.my\\_contributor](#).

### 6.20.3 Member Function Documentation

#### 6.20.3.1 void [umbra.editor.actions.BytecodeEditorAction.setActiveEditor](#) (final [IEditorPart a\\_part](#))

The method sets the bytecode [editor](#) for which the operation will be performed.

**Parameters:**

*a\_part* the bytecode [editor](#) to perform the operations

Reimplemented        in        [umbra.editor.actions.BytecodeColorAction](#),        and        [um-  
\[bra.editor.actions.BytecodeRefreshAction\]\(#\).](#)

References [umbra.editor.actions.BytecodeEditorAction.my\\_editor](#).

Referenced by [umbra.editor.BytecodeEditorContributor.setActiveEditor\(\)](#).

#### 6.20.3.2 final [BytecodeEditor umbra.editor.actions.BytecodeEditorAction.getEditor](#) ()

**Returns:**

the bytecode [editor](#) currently associated with the action

References [umbra.editor.actions.BytecodeEditorAction.my\\_editor](#).

Referenced        by        [umbra.editor.actions.BytecodeCombineAction.getClassPath\(\)](#),        um-  
[bra.editor.actions.BytecodeSynchrAction.getDocSynch\(\)](#), [umbra.editor.actions.history.BytecodeRestoreAction.getHistoryNum](#)  
[umbra.editor.actions.history.BytecodeRestoreAction.refreshContent\(\)](#),        um-  
[bra.editor.actions.BytecodeRebuildAction.replaceFile\(\)](#), [umbra.editor.actions.history.HistoryAction.run\(\)](#),  
[umbra.editor.actions.history.ClearHistoryAction.run\(\)](#), [umbra.editor.actions.history.BytecodeRestoreAction.run\(\)](#),  
[umbra.editor.actions.BytecodeSynchrAction.run\(\)](#),        [umbra.editor.actions.BytecodeRefreshAction.run\(\)](#),  
[umbra.editor.actions.BytecodeRebuildAction.run\(\)](#),        [umbra.editor.actions.BytecodeCombineAction.run\(\)](#),  
[umbra.editor.actions.BytecodeColorAction.run\(\)](#), [umbra.editor.actions.BytecodeRefreshAction setActiveEditor\(\)](#),  
[umbra.editor.actions.BytecodeCombineAction.updateMethods\(\)](#), [umbra.editor.actions.BytecodeCombineAction.updateMethod](#)  
and [umbra.editor.actions.BytecodeCombineAction.updateModifiedMethods\(\)](#).

#### 6.20.3.3 final [BytecodeEditorContributor um- \[bra.editor.actions.BytecodeEditorAction.getContributor\]\(#\) \(\)](#)

**Returns:**

the manager that initialises all the bytecode [actions](#) in the plugin

References [umbra.editor.actions.BytecodeEditorAction.my\\_contributor](#).

Referenced        by        [umbra.editor.actions.BytecodeRefreshAction.doRefresh\(\)](#),  
[umbra.editor.actions.history.BytecodeRestoreAction.refreshContent\(\)](#),        um-  
[bra.editor.actions.BytecodeCombineAction.refreshEditorWithClass\(\)](#),        um-  
[bra.editor.actions.BytecodeRebuildAction.run\(\)](#), and [umbra.editor.actions.BytecodeColorAction.run\(\)](#).

**6.20.3.4 final BytecodeContribution umbra.editor.actions.BytecodeEditorAction.getContribution  
()**

**Returns:**

the objects that encapsulates the GUI elements contributed by the bytecode plugin

References umbra.editor.actions.BytecodeEditorAction.my\_btcodeCntrbtn.

**6.20.3.5 static void umbra.editor.actions.BytecodeEditorAction.wrongLocationMessage (final  
Shell a\_shell, final String a\_title, final UmbraLocationException an\_ex) [static]**

Displays the message that a wrong location has been reached.

**Parameters:**

*a\_shell* the shell which displays the message

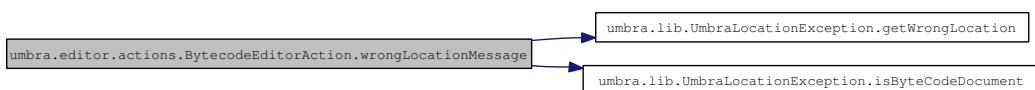
*a\_title* the title of the message window

*an\_ex* the exception with the information to display

References       umbra.lib.UmbraLocationException.getWrongLocation(),       and       umbra.lib.UmbraLocationException.isByteCodeDocument().

Referenced by umbra.editor.actions.BytecodeSynchrAction.run().

Here is the call graph for this function:



**6.20.3.6 static void umbra.editor.actions.BytecodeEditorAction.wrongFileOperationMessage (final  
Shell a\_shell, final String a\_title) [static]**

Displays the message that a file operation on a class file failed.

**Parameters:**

*a\_shell* the shell which displays the message

*a\_title* the title of the message window

Referenced       by       umbra.editor.actions.BytecodeRebuildAction.replaceFile(),       um-  
bra.editor.actions.BytecodeRefreshAction.run(),   umbra.editor.actions.BytecodeRebuildAction.run(),   um-  
bra.editor.actions.BytecodeCombineAction.run(), and umbra.editor.actions.BytecodeCombineAction.updateMethods().

**6.20.3.7 static void umbra.editor.actions.BytecodeEditorAction.wrongPathToClassMessage (final  
Shell a\_shell, final String a\_title, final String a\_path) [static]**

Displays the message that a given path does not lead to a valid class file.

**Parameters:**

- a\_shell* the shell which displays the message
- a\_title* the title of the message window
- a\_path* a path which was referenced

Referenced by umbra.editor.actions.BytecodeRefreshAction.run(), umbra.editor.actions.BytecodeRebuildAction.run(), and umbra.editor.actions.BytecodeCombineAction.updateMethods().

## 6.20.4 Member Data Documentation

### 6.20.4.1 BytecodeEditor umbra.editor.actions.BytecodeEditorAction.my\_editor [private]

The current byte code [editor](#) for which the action takes place.

Referenced by umbra.editor.actions.BytecodeCombineAction.getClassPath(), umbra.editor.actions.BytecodeEditorAction.getEditor(), umbra.editor.actions.BytecodeRefreshAction.run(), umbra.editor.actions.BytecodeEditorAction.setActiveEditor(), and umbra.editor.actions.BytecodeCombineAction.updateMethodsLogic().

### 6.20.4.2 BytecodeEditorContributor umbra.editor.actions.BytecodeEditorAction.my\_contributor [private]

The manager that initialises all the [actions](#) within the byte code plugin.

Referenced by umbra.editor.actions.BytecodeEditorAction.BytecodeEditorAction(), and umbra.editor.actions.BytecodeEditorAction.getContributor().

### 6.20.4.3 BytecodeContribution umbra.editor.actions.BytecodeEditorAction.my\_btcodeCntrbtn [private]

The GUI elements contributed to the eclipse GUI by the bytecode [editor](#).

This reference should be the same as in the field `my_contributor`.

Referenced by umbra.editor.actions.BytecodeEditorAction.BytecodeEditorAction(), and umbra.editor.actions.BytecodeEditorAction.getContribution().

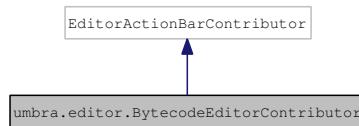
The documentation for this class was generated from the following file:

- source/umbra/editor/actions/[BytecodeEditorAction.java](#)

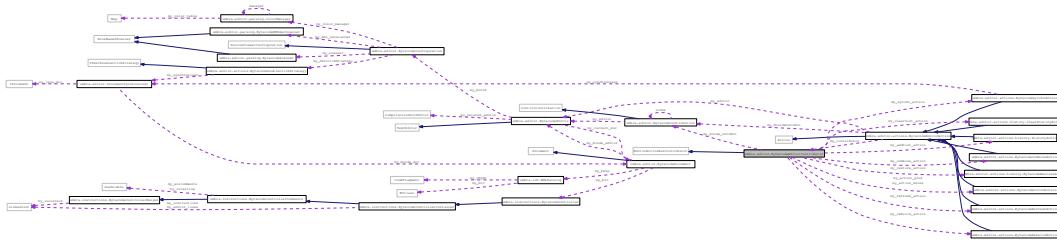
## 6.21 umbra.editor.BytecodeEditorContributor Class Reference

This is managing class that adds [actions](#) to workbench menus and toolbars for a byte code [editor](#).

Inheritance diagram for `umbra.editor.BytecodeEditorContributor`:



Collaboration diagram for `umbra.editor.BytecodeEditorContributor`:



### Public Member Functions

- `BytecodeEditorContributor ()`  
*The constructor is executed when the [editor](#) is started.*
- `final void contributeToToolBar (final IToolBarManager a_tbar_mngr)`  
*New buttons for the [actions](#) are added to the toolbar.*
- `final void contributeToMenu (final IMenuManager a_menu_mngr)`  
*The method creates a new menu with *Umbra* related items and adds the items to the menu.*
- `final void setActiveEditor (final IEditorPart an_editor)`  
*The current [editor](#) window is set as an attribute (also for each action).*
- `final BytecodeEditor refreshEditor (final BytecodeEditor an_editor, final String[ ] the_comments, final String[ ] the_interline) throws PartInitException, UmbraLocationException, UmbraMethodException`  
*The same as [IEditorInput, String\[\], String\[\]](#), but the input is obtained from the current [editor](#) window.*
- `final BytecodeEditor refreshEditor (final BytecodeEditor an_editor, final IEtherInput an_input, final String[ ] a_comment_array, final String[ ] an_interline) throws PartInitException, UmbraLocationException, UmbraMethodException`  
*Saves all settings of the current [editor](#) (selection positions, contributions, JavaClass structure, related [editor](#)).*
- `final BytecodeRefreshAction getRefreshAction ()`  
*debugging helper*

## Static Public Attributes

- static final String **REFRESH\_ID** = "umbra.editor.Refresh"

*The identifier of the refresh action.*

## Private Member Functions

- void **setupToolTipTexts** ()

*This method sets up the tool tip texts for all the **actions** except the colour mode **actions**.*

- void **createActions** ()

*This method creates the objects to handle all the **actions** except the colour mode **actions**.*

- void **setupColorActions** (final URL an\_install\_url, final int a\_mode)

*This method sets up all the **actions** which change the colouring style of the **editor**.*

- void **assignIcons** (final URL an\_install\_url)

*This method assigns appropriate icons to their respective **actions**.*

- void **wrongIconMessage** (final MalformedURLException an\_ex)

*The method pops up a message which informs that something is wrong with the paths to the Umbra icons.*

## Private Attributes

- **BytecodeContribution my\_bcode\_cntrbtn**

*The GUI element responsible for the communication between the GUI and the internal representation of a document.*

- **BytecodeColorAction my\_action\_plus**

*The action to change the colour mode to the next one.*

- **BytecodeColorAction my\_action\_minus**

*The action to change the colour mode to the previous one.*

- **BytecodeRefreshAction my\_refresh\_action**

*The action to refresh the content of the current byte code **editor**.*

- **BytecodeRebuildAction my\_rebuild\_action**

*The action to restore the original version of a class file.*

- **BytecodeCombineAction my\_combine\_action**

*The action to combine the modifications from the source code **editor** and from the byte code **editor**.*

- **HistoryAction my\_addhist\_action**

*The action to add one history snapshot.*

- **ClearHistoryAction my\_clearhist\_action**

*The action to clear all the history snapshots that were stored before.*

- [BytecodeRestoreAction my\\_restore\\_action](#)

*The action to restore one of the history snapshots that were stored before.*

- [BytecodeSynchrAction my\\_synchr\\_action](#)

*The action to synchronise the position in the byte code file with the corresponding position in the source code file.*

## 6.21.1 Detailed Description

This is managing class that adds [actions](#) to workbench menus and toolbars for a byte code [editor](#).

They appear when the [editor](#) is active. These [actions](#) are in particular: rebuild, refresh, combine, restore from history, synchronise the position of the cursor between the byte code and the Java code, colour change and check of the syntax correctness.

### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

### Version:

a-01

## 6.21.2 Constructor & Destructor Documentation

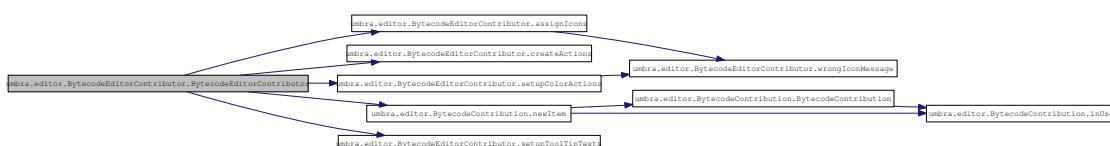
### 6.21.2.1 umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor ()

The constructor is executed when the [editor](#) is started.

This action happens when there is no byte code [editor](#) pane in the environment open and an action to open one is taken. This constructor creates all [actions](#) and provides them with their icons and tool tip texts. If necessary it assigns ids of the [actions](#).

References [umbra.editor.BytecodeEditorContributor.assignIcons\(\)](#), [umbra.editor.BytecodeEditorContributor.createActions\(\)](#), [umbra.editor.BytecodeEditorContributor.my\\_bcode\\_cntrbtn](#), [umbra.editor.BytecodeEditorContributor.my\\_refresh\\_action](#), [umbra.editor.BytecodeContribution.newItem\(\)](#), [umbra.editor.BytecodeEditorContributor.REFRESH\\_ID](#), [umbra.editor.BytecodeEditorContributor.setupColorActions\(\)](#), and [umbra.editor.BytecodeEditorContributor.setupToolTipTexts\(\)](#).

Here is the call graph for this function:



### 6.21.3 Member Function Documentation

#### 6.21.3.1 void umbra.editor.BytecodeEditorContributor.setupToolTipTexts () [private]

This method sets up the tool tip texts for all the [actions](#) except the colour mode [actions](#).

References [umbra.editor.BytecodeEditorContributor.my\\_addhist\\_action](#),  
[umbra.editor.BytecodeEditorContributor.my\\_clearhist\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_combine\\_action](#),  
[umbra.editor.BytecodeEditorContributor.my\\_rebuild\\_action](#),  
[umbra.editor.BytecodeEditorContributor.my\\_refresh\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_restore\\_action](#), and [umbra.editor.BytecodeEditorContributor.my\\_synchr\\_action](#).

Referenced by [umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor\(\)](#).

#### 6.21.3.2 void umbra.editor.BytecodeEditorContributor.createActions () [private]

This method creates the objects to handle all the [actions](#) except the colour mode [actions](#).

References [umbra.editor.BytecodeEditorContributor.my\\_addhist\\_action](#),  
[umbra.editor.BytecodeEditorContributor.my\\_bcode\\_cntrbtn](#), [umbra.editor.BytecodeEditorContributor.my\\_clearhist\\_action](#),  
[umbra.editor.BytecodeEditorContributor.my\\_combine\\_action](#),  
[umbra.editor.BytecodeEditorContributor.my\\_rebuild\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_refresh\\_action](#),  
[umbra.editor.BytecodeEditorContributor.my\\_restore\\_action](#), and  
[umbra.editor.BytecodeEditorContributor.my\\_synchr\\_action](#).

Referenced by [umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor\(\)](#).

#### 6.21.3.3 void umbra.editor.BytecodeEditorContributor.setupColorActions (final URL *an\_install\_url*, final int *a\_mode*) [private]

This method sets up all the [actions](#) which change the colouring style of the [editor](#).

##### Parameters:

- an\_install\_url* the URL to the location of the Umbra plugin installation
- a\_mode* the current colouring mode

References [umbra.editor.BytecodeEditorContributor.my\\_action\\_minus](#), [umbra.editor.BytecodeEditorContributor.my\\_action\\_plus](#), [umbra.editor.BytecodeEditorContributor.my\\_bcode\\_cntrbtn](#), and [umbra.editor.BytecodeEditorContributor.wrongIconMessage\(\)](#).

Referenced by [umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor\(\)](#).

Here is the call graph for this function:



#### 6.21.3.4 void umbra.editor.BytecodeEditorContributor.assignIcons (final URL *an\_install\_url*) [private]

This method assigns appropriate icons to their respective [actions](#).

**Parameters:**

*an\_install\_url* an URL to a location where the Umbra plugin is located

References [umbra.editor.BytecodeEditorContributor.my\\_addhist\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_clearhist\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_combine\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_rebuild\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_refresh\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_restore\\_action](#), [umbra.editor.BytecodeEditorContributor.my\\_synchr\\_action](#), and [umbra.editor.BytecodeEditorContributor.wrongIconMessage\(\)](#).

Referenced by [umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor\(\)](#).

Here is the call graph for this function:



#### 6.21.3.5 void umbra.editor.BytecodeEditorContributor.wrongIconMessage (final MalformedURLException *an\_ex*) [private]

The method pops up a message which informs that something is wrong with the paths to the Umbra icons.

**Parameters:**

*an\_ex* the exception for which the message should pop up

Referenced by [umbra.editor.BytecodeEditorContributor.assignIcons\(\)](#), and [umbra.editor.BytecodeEditorContributor.setupColorActions\(\)](#).

#### 6.21.3.6 final void umbra.editor.BytecodeEditorContributor.contributeToToolBar (final IToolBarManager *a\_tbar\_mngr*)

New buttons for the [actions](#) are added to the toolbar.

We call the superclass method and add:

- the refresh action icon
- the synchronisation icon

**Parameters:**

*a\_tbar\_mngr* the toolbar into which the widgets are added

**See also:**

[EditorActionBarContributor.contributeToToolBar\(IToolBarManager\)](#)

References [umbra.editor.BytecodeEditorContributor.my\\_refresh\\_action](#), and [umbra.editor.BytecodeEditorContributor.my\\_synchr\\_action](#).

### 6.21.3.7 final void umbra.editor.BytecodeEditorContributor.contributeToMenu (final IMenuManager *a\_menu\_mngr*)

The method creates a new menu with Umbra related items and adds the items to the menu.

**Parameters:**

*a\_menu\_mngr* the menu manager to add the Umbra menu to

References umbra.editor.BytecodeEditorContributor.my\_action\_minus, umbra.editor.BytecodeEditorContributor.my\_action\_plus, umbra.editor.BytecodeEditorContributor.my\_adhist\_action, umbra.editor.BytecodeEditorContributor.my\_clearhist\_action, umbra.editor.BytecodeEditorContributor.my\_combine\_action, umbra.editor.BytecodeEditorContributor.my\_rebuild\_action, umbra.editor.BytecodeEditorContributor.my\_refresh\_action, umbra.editor.BytecodeEditorContributor.my\_restore\_action, and umbra.editor.BytecodeEditorContributor.my\_synchr\_action.

### 6.21.3.8 final void umbra.editor.BytecodeEditorContributor.setActiveEditor (final IEeditorPart *an\_editor*)

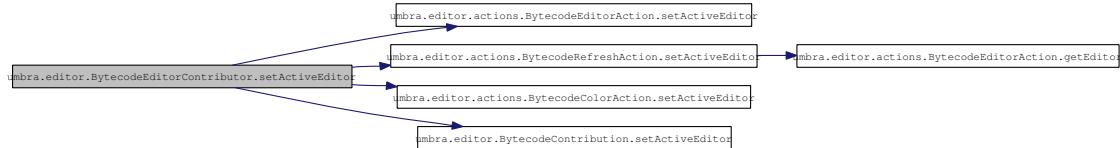
The current [editor](#) window is set as an attribute (also for each action).

**Parameters:**

*an\_editor* the current [editor](#) window

References umbra.editor.BytecodeEditorContributor.my\_action\_minus, umbra.editor.BytecodeEditorContributor.my\_action\_plus, umbra.editor.BytecodeEditorContributor.my\_adhist\_action, umbra.editor.BytecodeEditorContributor.my\_bcode\_cntrbtn, umbra.editor.BytecodeEditorContributor.my\_clearhist\_action, umbra.editor.BytecodeEditorContributor.my\_combine\_action, umbra.editor.BytecodeEditorContributor.my\_rebuild\_action, umbra.editor.BytecodeEditorContributor.my\_refresh\_action, umbra.editor.BytecodeEditorContributor.my\_restore\_action, umbra.editor.BytecodeEditorContributor.my\_synchr\_action, umbra.editor.BytecodeEditorContributor.REFRESH\_ID, umbra.editor.actions.BytecodeEditorAction setActiveEditor(), umbra.editor.actions.BytecodeRefreshAction setActiveEditor(), umbra.editor.actions.BytecodeColorAction setActiveEditor(), and umbra.editor.BytecodeContribution setActiveEditor().

Here is the call graph for this function:



### 6.21.3.9 final BytecodeEditor umbra.editor.BytecodeEditorContributor.refreshEditor (final BytecodeEditor *an\_editor*, final String[] *the\_comments*, final String[] *the\_interline*) throws PartInitException, UmbraLocationException, UmbraMethodException

The same as [IEeditorInput, String\[\], String\[\]](#), but the input is obtained from the current [editor](#) window.

**Parameters:**

*an\_editor* current [editor](#) to be closed

*the\_interline* an array with multi-line comments

*the\_comments* an array with end-of-line comments

#### Returns:

the new `editor`

#### Exceptions:

**PartInitException** if the new `editor` could not be created or initialised

**UmbralocationException** thrown in case a position has been reached which is outside the current document

**UmbramethodException** in case the textual representation has more methods than the internal one

#### See also:

[refreshEditor\(BytecodeEditor, IEditorInput, String\[\], String\[\]\)](#)

Referenced by `umbra.editor.actions.BytecodeRefreshAction.doRefresh()`,  
`umbra.editor.actions.history.BytecodeRestoreAction.refreshContent()`,  
`umbra.editor.actions.BytecodeCombineAction.refreshEditorWithClass()`,  
`umbra.editor.actions.BytecodeRebuildAction.run()`, and `umbra.editor.actions.BytecodeColorAction.run()`.

### 6.21.3.10 final BytecodeEditor umbra.editor.BytecodeEditorContributor.refreshEditor (final BytecodeEditor *an\_editor*, final IEditorInput *an\_input*, final String[ ] *a\_comment\_array*, final String[ ] *an\_interline*) throws PartInitException, UmbralocationException, UmbramethodException

Saves all settings of the current `editor` (selection positions, contributions, JavaClass structure, related `editor`).

Then closes the `editor` and opens a new one with the same settings and given input.

#### Parameters:

*an\_editor* current `editor` to be closed

*an\_input* input file to be displayed in new `editor`

*a\_comment\_array* contains the texts of end-of-line comments, the i-th entry contains the comment for the i-th instruction in the file, if this parameter is null then the array is not taken into account

*an\_interline* an array with multi-line comments //FIXME: currently ignored;  
<https://mobius.ucd.ie/ticket/555>

#### Returns:

the new `editor`

#### Exceptions:

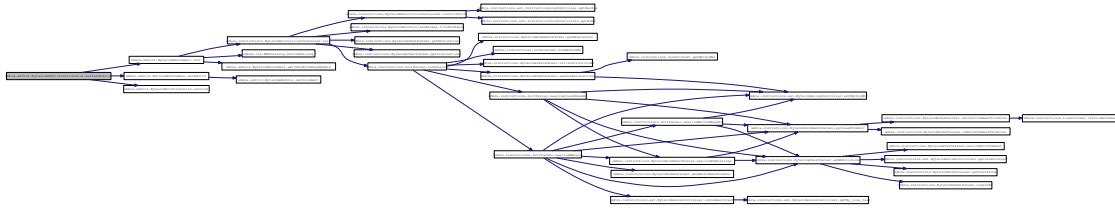
**PartInitException** if the new `editor` could not be created or initialised

**UmbralocationException** thrown in case a position has been reached which is outside the current document

**UmbramethodException** in case the textual representation has more methods than the internal one

References `umbra.editor.BytecodeDocument.init()`, `umbra.editor.BytecodeEditorContributor.my_bcode_cntrbtn`, `umbra.editor.BytecodeDocument.setEditor()`, and `umbra.editor.BytecodeContribution.survive()`.

Here is the call graph for this function:



### 6.21.3.11 final BytecodeRefreshAction `umbra.editor.BytecodeEditorContributor.getRefreshAction()`

debugging helper

```
/*private void controlPrint(JavaClass jc, int i) { Method meth = jc.getMethods()[i]; UmbraPlugin.messageLog(meth.getCode().toString()); }
```

**Returns:**

the action to refresh the byte code

References `umbra.editor.BytecodeEditorContributor.my_refresh_action`.

## 6.21.4 Member Data Documentation

### 6.21.4.1 final String `umbra.editor.BytecodeEditorContributor.REFRESH_ID = "umbra.editor.Refresh"` [static]

The identifier of the refresh action.

Referenced by `umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor()`, and `umbra.editor.BytecodeEditorContributor setActiveEditor()`.

### 6.21.4.2 BytecodeContribution `umbra.editor.BytecodeEditorContributor.my_bcode_cntrbtn` [private]

The GUI element responsible for the communication between the GUI and the internal representation of a document.

Referenced by `umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor()`, `umbra.editor.BytecodeEditorContributor.createActions()`, `umbra.editor.BytecodeEditorContributor.refreshEditor()`, `umbra.editor.BytecodeEditorContributor setActiveEditor()`, and `umbra.editor.BytecodeEditorContributor.setupColorActions()`.

### 6.21.4.3 BytecodeColorAction `umbra.editor.BytecodeEditorContributor.my_action_plus` [private]

The action to change the colour mode to the next one.

Referenced by umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupColorActions().

#### 6.21.4.4 BytecodeColorAction umbra.editor.BytecodeEditorContributor.my\_action\_minus [private]

The action to change the colour mode to the previous one.

Referenced by umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupColorActions().

#### 6.21.4.5 BytecodeRefreshAction umbra.editor.BytecodeEditorContributor.my\_refresh\_action [private]

The action to refresh the content of the current byte code [editor](#).

Referenced by umbra.editor.BytecodeEditorContributor.assignIcons(), umbra.editor.BytecodeEditorContributor.BytecodeEditorContributor(), umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.contributeToToolBar(), umbra.editor.BytecodeEditorContributor.createActions(), umbra.editor.BytecodeEditorContributor.getRefreshAction(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupToolTipTexts().

#### 6.21.4.6 BytecodeRebuildAction umbra.editor.BytecodeEditorContributor.my\_rebuild\_action [private]

The action to restore the original version of a class file.

Referenced by umbra.editor.BytecodeEditorContributor.assignIcons(), umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.createActions(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupToolTipTexts().

#### 6.21.4.7 BytecodeCombineAction umbra.editor.BytecodeEditorContributor.my\_combine\_action [private]

The action to combine the modifications from the source code [editor](#) and from the byte code [editor](#).

Referenced by umbra.editor.BytecodeEditorContributor.assignIcons(), umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.createActions(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupToolTipTexts().

#### 6.21.4.8 HistoryAction umbra.editor.BytecodeEditorContributor.my\_addhist\_action [private]

The action to add one history snapshot.

Referenced by umbra.editor.BytecodeEditorContributor.assignIcons(), umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.createActions(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupToolTipTexts().

**6.21.4.9 ClearHistoryAction umbra.editor.BytecodeEditorContributor.my\_clearhist\_action**  
[private]

The action to clear all the history snapshots that were stored before.

Referenced by umbra.editor.BytecodeEditorContributor.assignIcons(), umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.createActions(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupToolTipTexts().

**6.21.4.10 BytecodeRestoreAction umbra.editor.BytecodeEditorContributor.my\_restore\_action**  
[private]

The action to restore one of the history snapshots that were stored before.

Referenced by umbra.editor.BytecodeEditorContributor.assignIcons(), umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.createActions(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupToolTipTexts().

**6.21.4.11 BytecodeSynchrAction umbra.editor.BytecodeEditorContributor.my\_synchr\_action**  
[private]

The action to synchronise the position in the byte code file with the corresponding position in the source code file.

Referenced by umbra.editor.BytecodeEditorContributor.assignIcons(), umbra.editor.BytecodeEditorContributor.contributeToMenu(), umbra.editor.BytecodeEditorContributor.contributeToToolBar(), umbra.editor.BytecodeEditorContributor.createActions(), umbra.editor.BytecodeEditorContributor.setActiveEditor(), and umbra.editor.BytecodeEditorContributor.setupToolTipTexts().

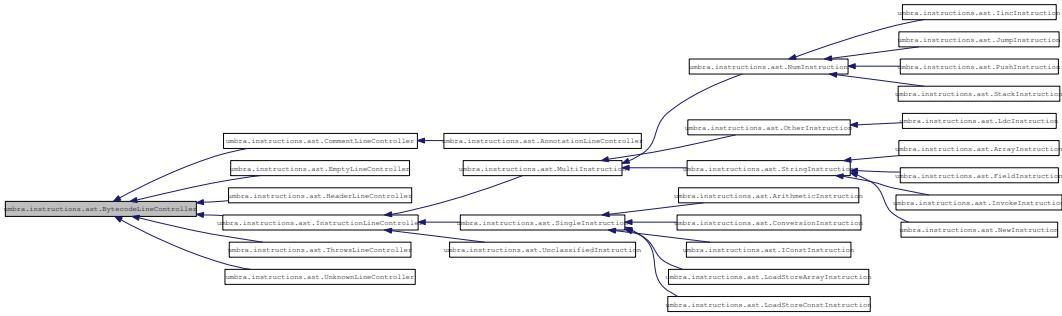
The documentation for this class was generated from the following file:

- source/umbra/editor/[BytecodeEditorContributor.java](#)

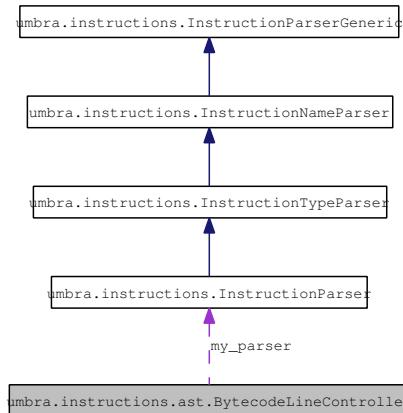
## 6.22 umbra.instructions.ast.BytecodeLineController Class Reference

This is completely abstract class that contains some information useful when the line is modified or BCEL structure is created.

Inheritance diagram for umbra.instructions.ast.BytecodeLineController:



Collaboration diagram for umbra.instructions.ast.BytecodeLineController:



## Public Member Functions

- **BytecodeLineController (final String a\_line)**  
*This constructor creates the controller with the given content of the line it handles.*
- boolean **addHandle** (final InstructionHandle an\_ihandle, final InstructionList a\_ilist, final MethodGen a\_methodgen)  
*The method adds the link between the Umbra representation of `instructions` to their representation in BCEL.*
- Instruction **getInstruction ()**  
*This method is redefined in each subclass of particular instruction type.*
- void **setTarget** (final InstructionList an\_ilist, final Instruction an\_ins) throws UmbraException  
*Sets the target of the given instruction.*

- `InstructionList getList ()`

*Returns the `InstructionList` structure in which the current instruction is located.*

- `MethodGen getMethod ()`

*Returns the `MethodGen` structure responsible for the method in which the instruction resides.*

- `final int getMethodNo ()`

*Return the method number of the method in which the line is located.*

- `boolean correct ()`

*This method is used to check some basic condition of correctness.*

- `void setMethodNo (final int an_index)`

*This method sets the number of the method which the current line belongs to.*

- `final String getLineContent ()`

*The method returns the String representation of the current instruction content.*

- `String getMy_line_text ()`

- `boolean isCommentEnd ()`

*Checks if the line can be an end of comment.*

- `boolean isCommentStart ()`

*Checks if the line can be an end of a comment.*

- `int getNoInMethod ()`

*This method returns the number of the instruction handled by the current line controller.*

- `boolean needsMg ()`

*Returns true when a BCEL method representation must be associated with the current line controller.*

- `boolean hasMg ()`

*Returns true when a BCEL method representation is associated with the current line controller.*

## Static Public Attributes

- `static final int WRONG_POSITION_IN_METHOD = -2`

*The constant returned that the byte code line cannot be assigned a meaningful position inside a method.*

## Protected Member Functions

- `InstructionParser getParser ()`

## Private Attributes

- **InstructionParser my\_parser**

*This is an object contains a parser which allows to check the correctness of the byte code line and to parse its parameters.*

- **int my\_methodno**

*The number of the method that contains the current line.*

- **String my\_line\_text**

*The string representation of the line in the byte code file that contains the current instruction.*

### 6.22.1 Detailed Description

This is completely abstract class that contains some information useful when the line is modified or BCEL structure is created.

Most details are implemented in subclasses.

Methods of this class should operate on the [org.apache.bcel.generic.ClassGen](#) object which is located in the [umbra.editor.ByticodeDocument](#) object that describes the state of the byte code [editor](#) which contains the line that corresponds to an object of the current class.

Note that some methods which logically belong to [InstructionLineController](#) are defined already here. This is caused by the fact that some of the line controllers may be associated with a method even though they do not handle [instructions](#) (but e.g. comments or empty lines).

#### Author:

Tomek Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.22.2 Constructor & Destructor Documentation

#### 6.22.2.1 umbra.instructions.ast.BytecodeLineController.BytecodeLineController (final String *a\_line*)

This constructor creates the controller with the given content of the line it handles.

It also creates the local parser which handles the parsing of the content of the line and initialises the association with a method so that no method is associated with the line controller.

#### Parameters:

*a\_line* the string representation of the line in the byte code document

References

umbra.instructions.ast.BytecodeLineController.my\_line\_text,  
umbra.instructions.ast.BytecodeLineController.my\_methodno, and  
umbra.instructions.ast.BytecodeLineController.my\_parser.

### 6.22.3 Member Function Documentation

#### 6.22.3.1 boolean `umbra.instructions.ast.BytecodeLineController.addHandle` (final `InstructionHandle an_ihandle`, final `InstructionList a_ilist`, final `MethodGen a_methodgen`)

The method adds the link between the Umbra representation of `instructions` to their representation in BCEL. In case the line does not correspond to an instruction we only register the number of the method the line is associated with.

**Parameters:**

`an_ihandle` the BCEL instruction handle that corresponds to the instruction associated with the current object  
`a_ilist` the list of `instructions` in the current method  
`a_methodgen` the object which represents the method of the current instruction in the BCEL representation of the current class in the byte code `editor`

**Returns:**

true when the current line corresponds to an instruction, false otherwise

Reimplemented in `umbra.instructions.ast.InstructionLineController`.

#### 6.22.3.2 Instruction `umbra.instructions.ast.BytecodeLineController.getInstruction` ()

This method is redefined in each subclass of particular instruction type.

It is used for creating a handle containing appropriate BCEL instruction object that matches with the instruction name.

**Returns:**

handle of the type `Instruction` to the appropriate instruction or `null` if the line is not an instruction one.

Reimplemented in `umbra.instructions.ast.ArithmeticInstruction`, `umbra.instructions.ast.ArrayInstruction`, `umbra.instructions.ast.ConversionInstruction`, `umbra.instructions.ast.FieldInstruction`, `umbra.instructions.ast.IConstInstruction`, `umbra.instructions.ast.IincInstruction`, `umbra.instructions.ast.InvokeInstruction`, `umbra.instructions.ast.JumpInstruction`, `umbra.instructions.ast.LdcInstruction`, `umbra.instructions.ast.LoadStoreArrayInstruction`, `umbra.instructions.ast.LoadStoreConstInstruction`, `umbra.instructions.ast.NewInstruction`, `umbra.instructions.ast.PushInstruction`, `umbra.instructions.ast.SingleInstruction`, and `umbra.instructions.ast.StackInstruction`.

Referenced by `umbra.instructions.ast.InstructionLineController.controlPrint()`, and `umbra.instructions.ast.InstructionLineController.replace()`.

#### 6.22.3.3 void `umbra.instructions.ast.BytecodeLineController.setTarget` (final `InstructionList an_ilist`, final `Instruction an_ins`) throws `UmbraException`

Sets the target of the given instruction.

This method is used to provide a common interface for all the `instructions`, but the actual work is done only in case of the jump `instructions`. Here it does nothing.

**Parameters:**

*an\_ilist* an instruction list with the jump instruction  
*an\_ins* the instruction to set the target for

**Exceptions:**

*UmbralException* when the instruction has improper target

**See also:**

[umbra.instructions.ast.BytecodeLineController.setTarget\(\)](#)      org.apache.bcel.generic.InstructionList,  
org.apache.bcel.genericInstruction)

Reimplemented in [umbra.instructions.ast.JumpInstruction](#).

#### 6.22.3.4 InstructionList [umbra.instructions.ast.BytecodeLineController.getList\(\)](#)

Returns the [InstructionList](#) structure in which the current instruction is located.

In case of [BytecodeLineController](#), this method always returns `null`.

**Returns:**

the BCEL list of the [instructions](#) of the method to which the current instruction belongs

Reimplemented in [umbra.instructions.ast.InstructionLineController](#).

#### 6.22.3.5 MethodGen [umbra.instructions.ast.BytecodeLineController.getMethod\(\)](#)

Returns the [MethodGen](#) structure responsible for the method in which the instruction resides.

In case of [BytecodeLineController](#) this method always returns `null`.

**Returns:**

the method in which the current instruction is located

Reimplemented in [umbra.instructions.ast.HeaderLineController](#), and [umbra.instructions.ast.InstructionLineController](#).

#### 6.22.3.6 final int [umbra.instructions.ast.BytecodeLineController.getMethodNo\(\)](#)

Return the method number of the method in which the line is located.

For lines that are not associated with any method this is equal to -1.

**Returns:**

method number

References [umbra.instructions.ast.BytecodeLineController.my\\_methodno](#).

Referenced by [umbra.instructions.BytecodeController.establishCurrentContext\(\)](#).

### 6.22.3.7 boolean [umbra.instructions.ast.BytecodeLineController.correct\(\)](#)

This method is used to check some basic condition of correctness.

For non-instruction line this is the only checking. It is usually redefined in the subclasses so that if it returns true the line is regarded to be correct.

**Returns:**

true if the instruction is correct

**See also:**

[InstructionLineController.correct\(\)](#)

Reimplemented in [umbra.instructions.ast.AnnotationLineController](#), [umbra.instructions.ast.ArithmeticInstruction](#), [umbra.instructions.ast.ArrayInstruction](#), [umbra.instructions.ast.CommentLineController](#), [umbra.instructions.ast.ConversionInstruction](#), [umbra.instructions.ast.EmptyLineController](#), [umbra.instructions.ast.FieldInstruction](#), [umbra.instructions.ast.HeaderLineController](#), [umbra.instructions.ast.IConstInstruction](#), [umbra.instructions.ast.IincInstruction](#), [umbra.instructions.ast.InstructionLineController](#), [umbra.instructions.ast.InvokeInstruction](#), [umbra.instructions.ast.JumpInstruction](#), [umbra.instructions.ast.LdcInstruction](#), [umbra.instructions.ast.LoadStoreArrayInstruction](#), [umbra.instructions.ast.LoadStoreConstInstruction](#), [umbra.instructions.ast.NewInstruction](#), [umbra.instructions.ast.PushInstruction](#), [umbra.instructions.ast.SingleInstruction](#), [umbra.instructions.ast.StackInstruction](#), [umbra.instructions.ast.ThrowsLineController](#), and [umbra.instructions.ast.UnclassifiedInstruction](#).

### 6.22.3.8 void [umbra.instructions.ast.BytecodeLineController.setMethodNo \(final int \*an\\_index\*\)](#)

This method sets the number of the method which the current line belongs to.

Normally, the number is not less than 0. The value -1 means the line is not associated with any method.

**Parameters:**

*an\_index* number of the method

References [umbra.instructions.ast.BytecodeLineController.my\\_methodno](#).

Referenced by [umbra.instructions.InitParser.swallowClassHeader\(\)](#), [umbra.instructions.InitParser.swallowMethod\(\)](#), [umbra.instructions.InitParser.swallowMethodHeader\(\)](#), and [umbra.instructions.BytecodeTextParser.updateAnnotations\(\)](#).

### 6.22.3.9 final String [umbra.instructions.ast.BytecodeLineController.getLineContent \(\)](#)

The method returns the String representation of the current instruction content.

**Returns:**

the representation of the line

References [umbra.instructions.ast.BytecodeLineController.my\\_line\\_text](#).

Referenced by [umbra.instructions.BytecodeTextParser.addEditorLine\(\)](#), and [umbra.instructions.BytecodeCommentParser.enrichWithComment\(\)](#).

**6.22.3.10 String umbra.instructions.ast.BytecodeLineController.getMy\_line\_text ()****Returns:**

the line of the text with the byte code line

References umbra.instructions.ast.BytecodeLineController.my\_line\_text.

Referenced by umbra.instructions.ast.HeaderLineController.correct(), umbra.instructions.ast.MultiInstruction.getInd(), umbra.instructions.ast.AnnotationLineController.isAnnotationEnd(), umbra.instructions.ast.CommentLineController.isCommentEnd(), and umbra.instructions.ast.BytecodeLineController.isCommentStart().

**6.22.3.11 InstructionParser umbra.instructions.ast.BytecodeLineController.getParser ()  
[protected]****Returns:**

the line parser for the current line

References umbra.instructions.ast.BytecodeLineController.my\_parser.

Referenced by umbra.instructions.ast.NumInstruction.checkInstructionWithNumber(), umbra.instructions.ast.StackInstruction.correct(), umbra.instructions.ast.PushInstruction.correct(), umbra.instructions.ast.NewInstruction.correct(), umbra.instructions.ast.LdcInstruction.correct(), umbra.instructions.ast.JumpInstruction.correct(), umbra.instructions.ast.InvokeInstruction.correct(), umbra.instructions.ast.IincInstruction.correct(), umbra.instructions.ast.FieldInstruction.correct(), umbra.instructions.ast.ArrayInstruction.correct(), umbra.instructions.ast.StackInstruction.getInd(), umbra.instructions.ast.PushInstruction.getInd(), umbra.instructions.ast.IincInstruction.getInd1(), umbra.instructions.ast.JumpInstruction.getInd(), umbra.instructions.ast.IincInstruction.getInd2(), umbra.instructions.ast.ArrayInstruction.getType(), and umbra.instructions.ast.InstructionLineController.parseTillMnemonic().

**6.22.3.12 boolean umbra.instructions.ast.BytecodeLineController.isCommentEnd ()**

Checks if the line can be an end of comment.

End of comment line can only be of [AnnotationLineController](#) type so the default behaviour is to always return false.

**Returns:**

`true` when the line contains the end of comment sequence, `false` otherwise

Reimplemented in [umbra.instructions.ast.CommentLineController](#).

**6.22.3.13 boolean umbra.instructions.ast.BytecodeLineController.isCommentStart ()**

Checks if the line can be an end of a comment.

**Returns:**

`true` when the line contains the end of comment sequence, `false` otherwise

References umbra.instructions.ast.BytecodeLineController.getMy\_line\_text().

Referenced by `umbra.instructions.InitParser.swallowMethod()`.

Here is the call graph for this function:



#### **6.22.3.14 int umbra.instructions.ast.BytecodeLineController.getNoInMethod ()**

This method returns the number of the instruction handled by the current line controller.

If no instruction can be associated with the line the value -2 is returned. In case of `BytecodeLineController`, this method always returns -2.

**Returns:**

the number of the instruction or -2 in case the number cannot be determined

References `umbra.instructions.ast.BytecodeLineController.WRONG_POSITION_IN_METHOD`.

#### **6.22.3.15 boolean umbra.instructions.ast.BytecodeLineController.needsMg ()**

Returns `true` when a BCEL method representation must be associated with the current line controller.

The default result is `false`.

**Returns:**

`true` when a BCEL method representation must be associated with the current line controller, otherwise `false`

Reimplemented in `umbra.instructions.ast.UnclassifiedInstruction`.

#### **6.22.3.16 boolean umbra.instructions.ast.BytecodeLineController.hasMg ()**

Returns `true` when a BCEL method representation is associated with the current line controller.

The default result is `false`.

**Returns:**

`true` when a BCEL method representation is associated with the current line controller, otherwise `false`

### **6.22.4 Member Data Documentation**

#### **6.22.4.1 final int umbra.instructions.ast.BytecodeLineController.WRONG\_POSITION\_IN\_- METHOD = -2 [static]**

The constant returned that the byte code line cannot be assigned a meaningful position inside a method.

Referenced by `umbra.instructions.ast.BytecodeLineController.getNoInMethod()`.

**6.22.4.2 InstructionParser umbra.instructions.ast.BytecodeLineController.my\_parser  
[private]**

This is an object contains a parser which allows to check the correctness of the byte code line and to parse its parameters.

Referenced by `umbra.instructions.ast.BytecodeLineController.BytecodeLineController()`, and `umbra.instructions.ast.BytecodeLineController.getParser()`.

**6.22.4.3 int umbra.instructions.ast.BytecodeLineController.my\_methodno [private]**

The number of the method that contains the current line.

This is an index in the `org.apache.bcel.generic.ClassGen` object available through the `umbra.editor.BytecodeDocument` object that describes the state of the byte code `editor` which contains the line that corresponds to the current object.

Values not less than zero mean the line is associated with a method. Values less than zero mean the line is not associated with any method.

Referenced by `umbra.instructions.ast.BytecodeLineController.BytecodeLineController()`, `umbra.instructions.ast.BytecodeLineController.getMethodNo()`, and `umbra.instructions.ast.BytecodeLineController.setMethodNo()`.

**6.22.4.4 String umbra.instructions.ast.BytecodeLineController.my\_line\_text [private]**

The string representation of the line in the byte code file that contains the current instruction.

We assume that the comments have been stripped off the line. The line text does not change in the lifetime of the object.

Referenced by `umbra.instructions.ast.BytecodeLineController.BytecodeLineController()`, `umbra.instructions.ast.MultiInstruction.getInd()`, `umbra.instructions.ast.BytecodeLineController.getLineContent()`, and `umbra.instructions.ast.BytecodeLineController.getMy_line_text()`.

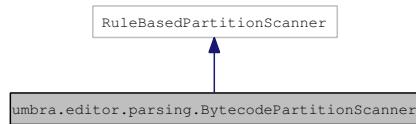
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[BytecodeLineController.java](#)

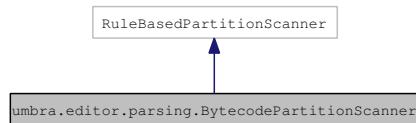
## 6.23 umbra.editor.parsing.BytecodePartitionScanner Class Reference

This class is responsible for dividing the byte code document into partitions the colouring of which is governed by different rules.

Inheritance diagram for umbra.editor.parsing.BytecodePartitionScanner:



Collaboration diagram for umbra.editor.parsing.BytecodePartitionScanner:



### Public Member Functions

- [BytecodePartitionScanner \(\)](#)

*This constructor creates rules and configures the scanner with them.*

### Static Public Attributes

- static final String [SECTION\\_HEAD](#) = "\_\_btc.header"

*This is the name of a content type assigned to areas of a byte code document that correspond to headers of methods or classes.*

- static final String [SECTION\\_THROWS](#) = "\_\_btc.throwsssec"

*This is the name of a content type assigned to areas of a byte code document that correspond to throws declarations.*

- static final String [SECTION\\_BML](#) = "\_\_btc.bmlcode"

*This is the name of a content type assigned to areas of a byte code document that correspond to BML annotations.*

### Static Private Attributes

- static final int [BML\\_RULE](#) = 0

*Index for the rule to handle BML annotations ending "@\*\\".*

- static final int [BML\\_RULE\\_SIMPLE](#) = 1

*Index for the rule to handle BML annotations ending "\*\\".*

- static final int **THROWS\_RULE** = 2

*Index for the rule to handle throws lines.*

- static final int **NUMBER\_OF\_RULES** = **THROWS\_RULE** + 1

*The total number of rules in the current scanner.*

### 6.23.1 Detailed Description

This class is responsible for dividing the byte code document into partitions the colouring of which is governed by different rules.

The text is divided into four kinds of regions:

- default section (governed by [BytecodeScanner](#)),
- section for headers (e.g. method, class or package headers; governed by [NonRuleBasedDamagerRepairer](#)),
- section for throws sections (governed by [NonRuleBasedDamagerRepairer](#)),
- section for BML annotations (governed by [BytecodeBMLSecScanner](#)).

#### Author:

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

Wojciech Was ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

### 6.23.2 Constructor & Destructor Documentation

#### 6.23.2.1 umbra.editor.parsing.BytecodePartitionScanner.BytecodePartitionScanner ()

This constructor creates rules and configures the scanner with them.

The rules handle the division of the byte code document into partitions the colouring of which is governed by different rules. The text is divided into four kinds of regions:

- default section,
- section for headers (e.g. method, class or package headers),
- section for throws sections,
- section for BML annotations.

References `umbra.editor.parsing.BytecodePartitionScanner.BML_RULE`,  
`umbra.editor.parsing.BytecodePartitionScanner.BML_RULE_SIMPLE`, `umbra.editor.parsing.BytecodePartitionScanner.NUMBER_OF_RULES`,  
`umbra.editor.parsing.BytecodePartitionScanner.SECTION_BML`, `umbra.editor.parsing.BytecodePartitionScanner.SECTION_HEAD`,  
`umbra.editor.parsing.BytecodePartitionScanner.SECTION_THROWS`, and  
`umbra.editor.parsing.BytecodePartitionScanner.THROWS_RULE`.

### 6.23.3 Member Data Documentation

#### 6.23.3.1 `final String umbra.editor.parsing.BytecodePartitionScanner.SECTION_HEAD = "__btc.header" [static]`

This is the name of a content type assigned to areas of a byte code document that correspond to headers of methods or classes.

These include lines with comments, lines with public declarations, lines with private declarations, lines with protected declarations, lines with braces, and lines with class declarations.

Referenced by `umbra.editor.parsing.BytecodePartitionScanner.BytecodePartitionScanner()`.

#### 6.23.3.2 `final String umbra.editor.parsing.BytecodePartitionScanner.SECTION_THROWS = "__btc.throwssec" [static]`

This is the name of a content type assigned to areas of a byte code document that correspond to throws declarations.

FIXME: the handling of these sections is partial; <https://mobius.ucd.ie/ticket/549>

Referenced by `umbra.editor.parsing.BytecodePartitionScanner.BytecodePartitionScanner()`.

#### 6.23.3.3 `final String umbra.editor.parsing.BytecodePartitionScanner.SECTION_BML = "__btc.bmlcode" [static]`

This is the name of a content type assigned to areas of a byte code document that correspond to BML annotations.

Referenced by `umbra.editor.parsing.BytecodePartitionScanner.BytecodePartitionScanner()`.

#### 6.23.3.4 `final int umbra.editor.parsing.BytecodePartitionScanner.BML_RULE = 0 [static, private]`

Index for the rule to handle BML annotations ending "@\*\\".

Referenced by `umbra.editor.parsing.BytecodePartitionScanner.BytecodePartitionScanner()`.

#### 6.23.3.5 `final int umbra.editor.parsing.BytecodePartitionScanner.BML_RULE_SIMPLE = 1 [static, private]`

Index for the rule to handle BML annotations ending "\*\".

Referenced by `umbra.editor.parsing.BytecodePartitionScanner.BytecodePartitionScanner()`.

**6.23.3.6 final int umbra.editor.parsing.BytecodePartitionScanner.THROWS\_RULE = 2**  
[static, private]

Index for the rule to handle throws lines.

Referenced by `umbra.editor.parsing.BytecodePartitionScanner.BytecodePartitionScanner()`.

**6.23.3.7 final int umbra.editor.parsing.BytecodePartitionScanner.NUMBER\_OF\_RULES = THROWS\_RULE + 1** [static, private]

The total number of rules in the current scanner.

It is by one greater than the maximal rule number.

Referenced by `umbra.editor.parsing.BytecodePartitionScanner.BytecodePartitionScanner()`.

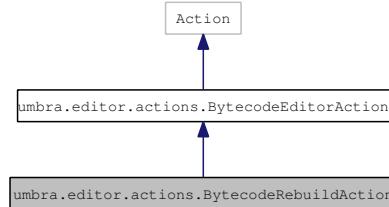
The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/[BytecodePartitionScanner.java](#)

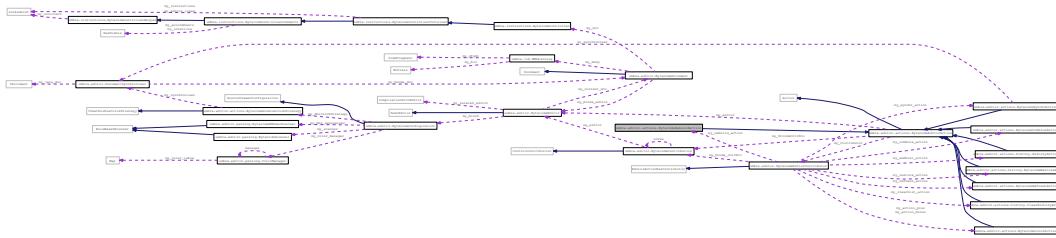
## 6.24 umbra.editor.actions.BytecodeRebuildAction Class Reference

This class defines action of restoring the original version of a class file (it is saved with the name prefixed with '\_') and then generating byte code (.btc) directly from it.

Inheritance diagram for umbra.editor.actions.BytecodeRebuildAction:



Collaboration diagram for umbra.editor.actions.BytecodeRebuildAction:



### Public Member Functions

- [BytecodeRebuildAction](#) (final [BytecodeEditorContributor](#) a\_contributor, final [BytecodeContribution](#) a\_bytecode\_contribution)

*This constructor creates the action to restore the original contents of the class file.*

- final void [run](#) ()

*This method restores a saved copy of the original .class file that resulted from the Java source file (it is stored under the name of the class file prefixed with '\_').*

### Private Member Functions

- void [replaceFile](#) (final [IFile](#) a\_filefrom, final [IPath](#) a\_pathto)

*The method replaces file at the path pathTo with the file determined by fileFrom.*

#### 6.24.1 Detailed Description

This class defines action of restoring the original version of a class file (it is saved with the name prefixed with '\_') and then generating byte code (.btc) directly from it.

In this way, all the changes made up to now are removed.

**Author:**

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
 Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
 Wojciech Was ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

**Version:**

a-01

**6.24.2 Constructor & Destructor Documentation****6.24.2.1 umbra.editor.actions.BytecodeRebuildAction(BytecodeRebuildAction final BytecodeEditorContributor a\_contributor, final BytecodeContribution a\_bytecode\_contribution)**

This constructor creates the action to restore the original contents of the class file.

It registers the name of the action with the text "Rebuild" and stores locally the object which creates all the [actions](#) and which contributes the [editor](#) GUI elements to the eclipse GUI.

**Parameters:**

*a\_contributor* the manager that initialises all the [actions](#) within the byte code plugin

*a\_bytecode\_contribution* the GUI elements contributed to the eclipse GUI by the byte code [editor](#).  
 This reference should be the same as in the parameter *a\_contributor*.

**6.24.3 Member Function Documentation****6.24.3.1 final void umbra.editor.actions.BytecodeRebuildAction.run ()**

This method restores a saved copy of the original .class file that resulted from the Java source file (it is stored under the name of the class file prefixed with '\_').

The class file with our modifications is removed, and the [editor](#) input is updated together with the [editor](#) window.

References [umbra.editor.actions.BytecodeEditorAction.getContributor\(\)](#), [umbra.editor.BytecodeEditor.getDocument\(\)](#), [umbra.editor.actions.BytecodeEditorAction.getEditor\(\)](#), [umbra.editor.BytecodeEditorContributor.refreshEditor\(\)](#), [umbra.editor.actions.BytecodeRebuildAction.replaceFile\(\)](#), [umbra.editor.actions.BytecodeEditorAction.wrongFileOperationMessage\(\)](#), and [umbra.editor.actions.BytecodeEditorAction.wrongPathToClassMessage\(\)](#).

Here is the call graph for this function:



### 6.24.3.2 void umbra.editor.actions.BytecodeRebuildAction.replaceFile (final IFile *a\_filefrom*, final IPath *a\_pathto*) [private]

The method replaces file at the path *pathTo* with the file determined by *fileFrom*.

This method pops up a message in case the operation cannot be executed.

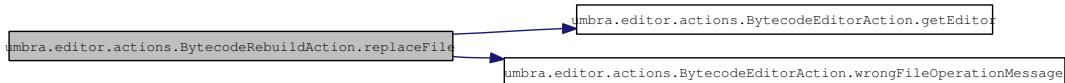
#### Parameters:

- a\_filefrom* a file which replaces
- a\_pathto* a location to be replaced

References      [umbra.editor.actions.BytecodeEditorAction.getEditor\(\)](#),      and      [umbra.editor.actions.BytecodeEditorAction.wrongFileOperationMessage\(\)](#).

Referenced by [umbra.editor.actions.BytecodeRebuildAction.run\(\)](#).

Here is the call graph for this function:



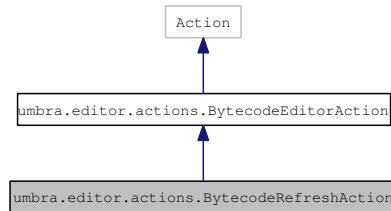
The documentation for this class was generated from the following file:

- [source/umbra/editor/actions/BytecodeRebuildAction.java](#)

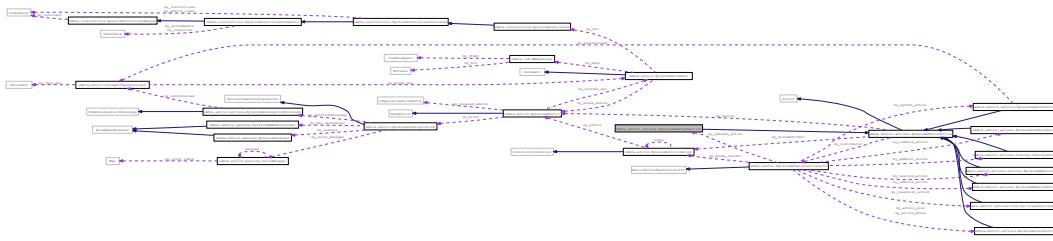
## 6.25 umbra.editor.actions.BytecodeRefreshAction Class Reference

This is a class defining an action: save current byte code `editor` window and re-generate byte code from the `.class` file.

Inheritance diagram for `umbra.editor.actions.BytecodeRefreshAction`:



Collaboration diagram for `umbra.editor.actions.BytecodeRefreshAction`:



### Public Member Functions

- `BytecodeRefreshAction (final BytecodeEditorContributor a_contributor, final BytecodeContribution a_bytecode_contribution)`

*This constructor creates the action to refresh the byte code `editor`.*

- `final void setActiveEditor (final IEditorPart a_target_editor)`

*This method sets the byte code `editor` for which the refresh action will be executed.*

- `final void run ()`

*This method saves the `editor` content in the files `.btc`.*

### Private Member Functions

- `BytecodeEditor doRefresh (final BytecodeEditor the_editor, final IFile a_file) throws ClassNotFoundException, CoreException, UmbraRangeException`

*This method does the actual job of refreshing the content of the byte code `editor` with an already saved content of a class file.*

### 6.25.1 Detailed Description

This is a class defining an action: save current byte code `editor` window and re-generate byte code from the `.class` file.

This action is equivalent to the generation of the byte code again from the Java code after saving binary file.

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
 Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.25.2 Constructor & Destructor Documentation

#### 6.25.2.1 `umbra.editor.actions.BytecodeRefreshAction` (final BytecodeEditorContributor *a\_contributor*, final BytecodeContribution *a\_bytecode\_contribution*)

This constructor creates the action to refresh the byte code `editor`.

It registers the name of the action with the text "Refresh" and stores locally the object which creates all the `actions` and which contributes the `editor` GUI elements to the eclipse GUI.

**Parameters:**

*a\_contributor* the manager that initialises all the `actions` within the byte code plugin  
*a\_bytecode\_contribution* the GUI elements contributed to the eclipse GUI by the byte code `editor`.  
 This reference should be the same as in the parameter *a\_contributor*.

### 6.25.3 Member Function Documentation

#### 6.25.3.1 `final void umbra.editor.actions.BytecodeRefreshAction setActiveEditor` (final IEeditorPart *a\_target\_editor*)

This method sets the byte code `editor` for which the refresh action will be executed.

Except for the superclass functionality it sets the refresh action to be active in case the `editor` is dirty.

**Parameters:**

*a\_target\_editor* the byte code `editor` for which the action will be executed

Reimplemented from `umbra.editor.actions.BytecodeEditorAction`.

References `umbra.editor.actions.BytecodeEditorAction.getEditor()`.

Referenced by `umbra.editor.actions.BytecodeRefreshAction.run()`, and `umbra.editor.BytecodeEditorContributor.setActiveEditor()`.

Here is the call graph for this function:



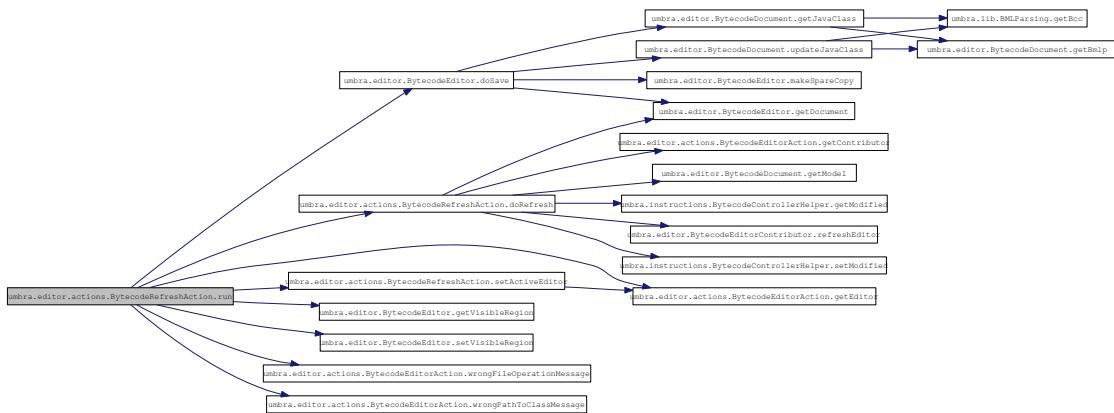
### 6.25.3.2 final void umbra.editor.actions.BytecodeRefreshAction.run ()

This method saves the [editor](#) content in the files .btc.

and .class associated with it and then creates a new input from the .class file. Finally replaces content of the [editor](#) window with the newly generated text. The general idea is that the current modifications are stored in a file and then retrieved back to the [editor](#) to obtain a nicely formatted presentation of the code.

References                [umbra.editor.actions.BytecodeRefreshAction.doRefresh\(\)](#),                um-  
[bra.editor.BytecodeEditor.doSave\(\)](#),                [umbra.editor.actions.BytecodeEditorAction.getEditor\(\)](#),                um-  
[bra.editor.BytecodeEditor.getVisibleRegion\(\)](#),                [umbra.editor.actions.BytecodeEditorAction.my\\_](#)  
[editor](#),                [umbra.editor.actions.BytecodeRefreshAction.setActiveEditor\(\)](#),                um-  
[bra.editor.BytecodeEditor.setVisibleRegion\(\)](#), [umbra.editor.actions.BytecodeEditorAction.wrongFileOperationMessage\(\)](#),  
and [umbra.editor.actions.BytecodeEditorAction.wrongPathToClassMessage\(\)](#).

Here is the call graph for this function:



### 6.25.3.3 BytecodeEditor umbra.editor.actions.BytecodeRefreshAction.doRefresh (final BytecodeEditor *the\_editor*, final IFile *a\_file*) throws ClassNotFoundException, CoreException, UmbraRangeException [private]

This method does the actual job of refreshing the content of the byte code [editor](#) with an already saved content of a class file.

First, we obtain the path to the class file. Then we store temporarily the comments and information on the modified methods. Then we refresh the byte code and the [editor](#).

#### Parameters:

*the\_editor* the [editor](#) which is refreshed  
*a\_file* the .btc file the content of which is refreshed

#### Returns:

the fresh [editor](#)

#### Exceptions:

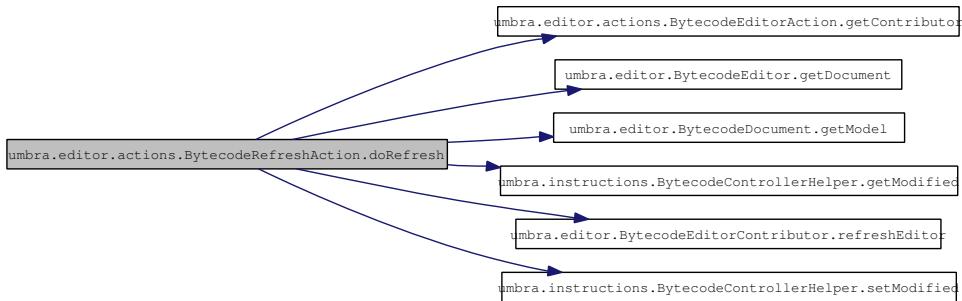
**ClassNotFoundException** the class corresponding to the given file cannot be found  
**CoreException** a file operation on the byte code file did not succeed

***UmbralRangeException*** thrown in case a position has been reached which is outside the current document or when the textual representation has more methods than the internal one

References      `umbra.editor.actions.BytecodeEditorAction.getContributor()`,  
`umbra.editor.BytecodeEditor.getDocument()`,      `umbra.editor.BytecodeDocument.getModel()`,  
`umbra.instructions.BytecodeControllerHelper.getModified()`, `umbra.editor.BytecodeEditorContributor.refreshEditor()`,  
and `umbra.instructions.BytecodeControllerHelper.setModified()`.

Referenced by `umbra.editor.actions.BytecodeRefreshAction.run()`.

Here is the call graph for this function:



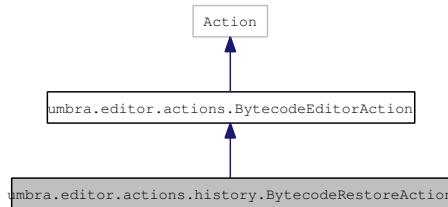
The documentation for this class was generated from the following file:

- source/umbra/editor/actions/[BytecodeRefreshAction.java](#)

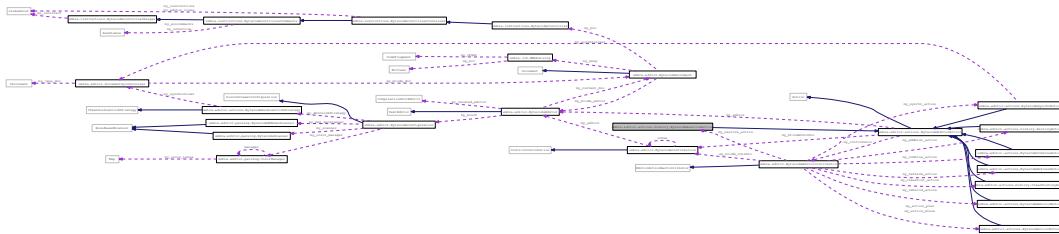
## 6.26 umbra.editor.actions.history.BytecodeRestoreAction Class Reference

This class defines action of restoring byte code from [history](#).

Inheritance diagram for umbra.editor.actions.history.BytecodeRestoreAction:



Collaboration diagram for umbra.editor.actions.history.BytecodeRestoreAction:



### Public Member Functions

- `BytecodeRestoreAction (final BytecodeEditorContributor a_contributor, final BytecodeContribution a_btcd_contribution)`

*This constructor creates the action to restore a file stored in the [history](#) of the bytecode editor.*

- `final void run ()`

*This method prompts the user for a [history](#) item number and restores the corresponding [history](#) item.*

### Private Member Functions

- `void refreshContent (final BytecodeEditor an_editor, final IFile a_btcffile, final IFile a_classfile)`  
throws CoreException

*The operation to refresh the content of the given [editor](#) from the class file is performed.*

- `int getHistoryNum ()`

*This method asks the user to give the [history](#) version number.*

### 6.26.1 Detailed Description

This class defines action of restoring byte code from [history](#).

Current version is replaced with one of these kept in [history](#) as a file with bt1, bt2, etc. extensions.

**Author:**

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
 Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
 Wojciech Was ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
 Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

## 6.26.2 Constructor & Destructor Documentation

### 6.26.2.1 `umbra.editor.actions.history.BytecodeRestoreAction`. `BytecodeRestoreAction` `(final BytecodeEditorContributor a_contributor, final BytecodeContribution a_btcd_contribution)`

This constructor creates the action to restore a file stored in the [history](#) of the bytecode [editor](#).

It registers the name of the action with the text "Restore" and stores locally the object which creates all the [actions](#) and which contributes the [editor](#) GUI elements to the eclipse GUI.

**Parameters:**

*a\_contributor* the manager that initialises all the [actions](#) within the bytecode plugin

*a\_btcd\_contribution* the GUI elements contributed to the eclipse GUI by the bytecode [editor](#). This reference should be the same as in the parameter *a\_contributor*.

## 6.26.3 Member Function Documentation

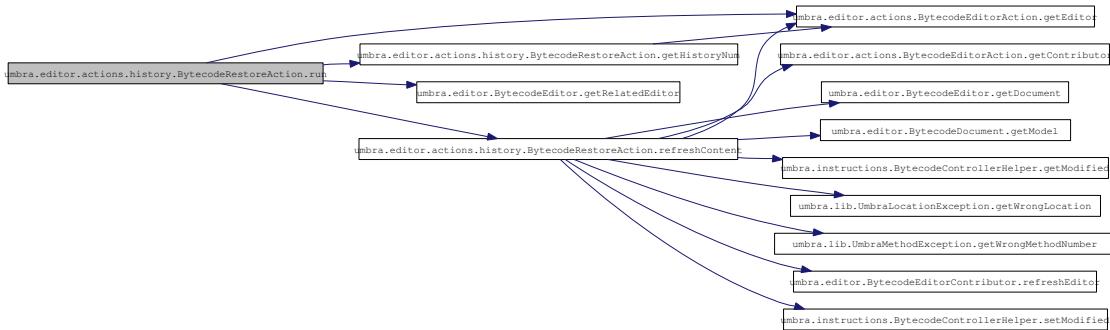
### 6.26.3.1 `final void umbra.editor.actions.history.BytecodeRestoreAction.run ()`

This method prompts the user for a [history](#) item number and restores the corresponding [history](#) item.

An input dialog to insert number of version to restore is shown. Then the current working class file name is established and corresponding .btc [history](#) file is loaded and .class file is loaded. At last the content of the .btc file is refreshed and the synchronisation action is enabled. In case an error is encountered, an appropriate message is displayed.

References	<code>umbra.editor.actions.BytecodeEditorAction.getEditor()</code> ,	<code>um-</code>
	<code>umbra.editor.actions.history.BytecodeRestoreAction.getHistoryNum()</code> ,	<code>um-</code>
	<code>umbra.editor.BytecodeEditor.getRelatedEditor()</code> , and <code>umbra.editor.actions.history.BytecodeRestoreAction.refreshContent()</code> .	

Here is the call graph for this function:



### 6.26.3.2 void umbra.editor.actions.history.BytecodeRestoreAction.refreshContent (final BytecodeEditor *an\_editor*, final IFile *a\_btcf file*, final IFile *a\_classfile*) throws CoreException [private]

The operation to refresh the content of the given [editor](#) from the class file is performed.

In case of an error an appropriate message is displayed.

#### Parameters:

*an\_editor* the [editor](#) in which the refresh operation is done

*a\_btcf file* the .btc file for which the refresh operation is done

*a\_classfile* the class file corresponding to the .btc file

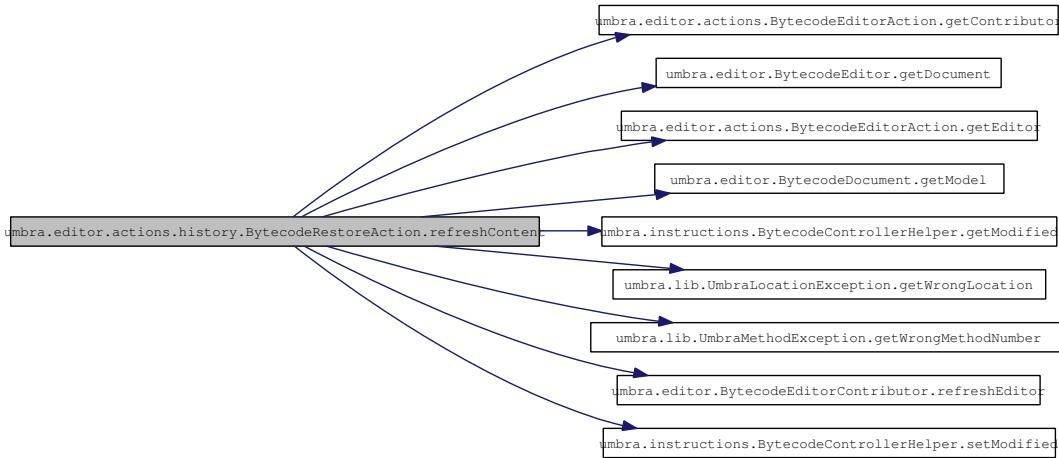
#### Exceptions:

[CoreException](#) in case the file system operations cannot be performed

References      [umbra.editor.actions\[BytecodeEditorAction\].getContributor\(\)](#), [umbra.editor\[BytecodeEditor\].getDocument\(\)](#), [umbra.editor.actions\[BytecodeEditorAction\].getEditor\(\)](#), [umbra.editor\[BytecodeDocument\].getModel\(\)](#), [umbra.instructions\[BytecodeControllerHelper\].getModified\(\)](#), [umbra.lib\[UmbraLocationException\].getWrongLocation\(\)](#), [umbra.lib\[UmbraMethodException\].getWrongMethodNumber\(\)](#), [umbra.editor\[BytecodeEditorContributor\].refreshEditor\(\)](#), and [umbra.instructions\[BytecodeControllerHelper\].setModified\(\)](#).

Referenced by [umbra.editor.actions.history.BytecodeRestoreAction.run\(\)](#).

Here is the call graph for this function:



### 6.26.3.3 int umbra.editor.actions.history.BytecodeRestoreAction.getHistoryNum () [private]

This method asks the user to give the [history](#) version number.

In case the given value is not a number or is a number outside of the range [HistoryOperations#MIN\\_HISTORY](#)-[HistoryOperations#MAX\\_HISTORY](#) the method asks to confirm the default value ([HistoryOperations#DEFAULT\\_HISTORY](#)). The user can refuse to accept the default and then the procedure repeats.

#### Returns:

the [history](#) item number given by the user

References `umbra.editor.actions.BytecodeEditorAction.getEditor()`.

Referenced by `umbra.editor.actions.history.BytecodeRestoreAction.run()`.

Here is the call graph for this function:



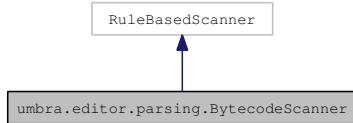
The documentation for this class was generated from the following file:

- [source/umbra/editor/actions/history/BytecodeRestoreAction.java](#)

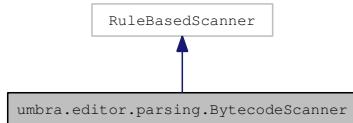
## 6.27 umbra.editor.parsing.BytecodeScanner Class Reference

This class is responsible for colouring these texts in a byte code editor window which are outside BML annotations areas.

Inheritance diagram for umbra.editor.parsing.BytecodeScanner:



Collaboration diagram for umbra.editor.parsing.BytecodeScanner:



### Public Member Functions

- BytecodeScanner (final ColorManager the\_manager, final int a\_mode)

*The constructor initialises the scanning rules for the current scanner.*

### Private Member Functions

- IRule[ ] createRulesArray (final IToken[ ] the\_tokens)

*The method creates an array of rules that are used in the colouring of an edited byte code document.*

- WordRule createInstructionRule (final IToken[ ] the\_tokens)

*This method creates a rule used for colouring various kinds of words that occur in a byte code document.*

- void tokensForCodeline (final IToken[ ] the\_tokens, final WordRule the\_insrule)

*This method associates in the\_insrule the words which occur in a line with the "Code" keyword with the token in the\_tokens under ColorValues#SLOT\_LINE.*

- void tokensForLinewords (final IToken[ ] the\_tokens, final WordRule the\_insrule)

*This method associates in the\_insrule the words which occur in a line with the "Line" keyword with the token in the\_tokens under ColorValues#SLOT\_LINE.*

- void tokensForInstructions (final IToken[ ] the\_tokens, final WordRule the\_insrule)

*This method associates in the\_insrule the words of the byte code instructions with the token in the\_tokens under ColorValues#SLOT\_BTCTINSTR.*

## Static Private Attributes

- static final int **RULE\_EOL** = 0  
*The number of the rule which is responsible for colour and text styling of the end-of-line comments.*
- static final int **RULE\_WORDS** = 1  
*The number of the rule which is responsible for colour and text styling of the words (as defined in [Bytecode-WordDetector](#)).*
- static final int **RULE\_INS\_NUMBER** = 2  
*The number of the rule which is responsible for colour and text styling of the byte code instruction numbers at the beginning of a method line.*
- static final int **RULE\_HASH** = 3  
*The number of the rule which is responsible for colour and text styling of the numbers preceded by the hash (#) sign.*
- static final int **RULE\_PERCENT** = 4  
*The number of the rule which is responsible for colour and text styling of the numbers preceded by the percent (%) sign.*
- static final int **RULE\_PARENTHESSES** = 5  
*The number of the rule which is responsible for colour and text styling of the numbers enclosed in the parentheses ('(', ')').*
- static final int **RULE\_BRACES** = 6  
*The number of the rule which is responsible for colour and text styling of the line parts enclosed in the braces ('{', '}').*
- static final int **RULE\_NUMBER** = 7  
*The number of the rule which is responsible for colour and text styling of the numbers (without #, %, or surrounding parentheses).*
- static final int **RULE\_WHITESPACE** = 8  
*The number of the rule which is responsible for colour and text styling of the whitespace areas.*
- static final int **RULE\_COMMENT** = 9  
*The number of the rule which is responsible for colour and text styling of comments.*
- static final int **RULE\_ANNOT** = 10  
*The number of the rule which is responsible for colour and text styling of BML annotations areas ending with @\*/\|.*
- static final int **RULE\_ANNOT\_SIMPLE** = 11  
*The number of the rule which is responsible for colour and text styling of BML annotations areas ending with \*\|.*
- static final int **NUMBER\_OF\_RULES** = **RULE\_ANNOT\_SIMPLE** + 1  
*The number of colouring rules used in this class.*

### 6.27.1 Detailed Description

This class is responsible for colouring these texts in a byte code editor window which are outside BML annotations areas.

This class uses special 10 rules which describe the way the different areas are coloured. Colours are chosen as a token array with a particular colouring style given in the constructor.

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
 Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.27.2 Constructor & Destructor Documentation

#### 6.27.2.1 umbra.editor.parsing.BytecodeScanner(BytecodeScanner (final ColorManager *the\_manager*, final int *a\_mode*)

The constructor initialises the scanning rules for the current scanner.

It creates and the scanning rules taking into account the given colour manager and colouring mode. It creates the rules to recognise all the 10 rules:

- end-of-line comments,
- words,
- instruction number labels,
- numbers starting with '#',
- numbers starting with ",
- numbers in parentheses '(', ')',
- line sections in braces '{', '}',
- bare numbers,
- whitespace,
- comments.

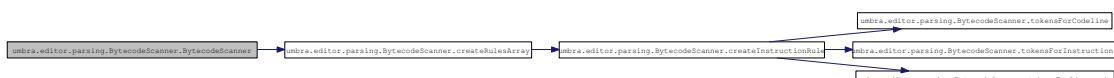
**Parameters:**

*the\_manager* the colour manager related to the current byte code editor, it must be the same as in the current [umbra.editor.BytecodeConfiguration](#) object

*a\_mode* the number of the current colouring style, it must be the same as in the current [umbra.editor.BytecodeConfiguration](#) object

References [umbra.editor.parsing.BytecodeScanner.createRulesArray\(\)](#).

Here is the call graph for this function:



### 6.27.3 Member Function Documentation

#### 6.27.3.1 IRule [ ] `umbra.editor.parsing.BytecodeScanner.createRulesArray (final IToken[ ] the_tokens)` [private]

The method creates an array of rules that are used in the colouring of an edited byte code document.

The array has the size `NUMBER_OF_RULES`} and its elements are filled as the descriptions of the constants RULE\_\*

- `RULE_EOL` for end-of-line comments,
- `RULE_WORDS` for words,
- `RULE_INS_NUMBER` for instruction number labels,
- `RULE_HASH` for numbers starting with '#',
- `RULE_PERCENT` for numbers starting with %,
- `RULE_PARENTHESSES` for numbers in parentheses '(', ')',
- `RULE_BRACES` for line sections in braces '{', '}',
- `RULE_NUMBER` for bare numbers,
- `RULE_WHITESPACE` for whitespace,
- `RULE_COMMENT` for comments.

#### Parameters:

`the_tokens` the array of tokens that are returned when rules are applied

#### Returns:

the array with the rules

References `umbra.editor.parsing.BytecodeScanner.createInstructionRule()`,  
`umbra.editor.parsing.BytecodeScanner.NUMBER_OF_RULES`, `umbra.editor.parsing.BytecodeScanner.RULE_ANNOT`,  
`umbra.editor.parsing.BytecodeScanner.RULE_ANNOT_SIMPLE`,  
`umbra.editor.parsing.BytecodeScanner.RULE_BRACES`, `umbra.editor.parsing.BytecodeScanner.RULE_COMMENT`, `umbra.editor.parsing.BytecodeScanner.RULE_EOL`, `umbra.editor.parsing.BytecodeScanner.RULE_HASH`, `umbra.editor.parsing.BytecodeScanner.RULE_INS_NUMBER`, `umbra.editor.parsing.BytecodeScanner.RULE_NUMBER`,  
`umbra.editor.parsing.BytecodeScanner.RULE_PARENTHESES`,  
`umbra.editor.parsing.BytecodeScanner.RULE_PERCENT`, `umbra.editor.parsing.BytecodeScanner.RULE_WHITESPACE`, and `umbra.editor.parsing.BytecodeScanner.RULE_WORDS`.

Referenced by `umbra.editor.parsing.BytecodeScanner.BytecodeScanner()`.

Here is the call graph for this function:



### 6.27.3.2 WordRule umbra.editor.parsing.BytecodeScanner.createInstructionRule (final IToken[ ] *the\_tokens*) [private]

This method creates a rule used for colouring various kinds of words that occur in a byte code document.

It assigns the [ColorValues#SLOT\\_DEFAULT](#) colour as the default colour for words. Except for that it assigns special colouring rules for the word categories: the byte code [instructions](#), keywords in a "Line" section, and keywords in the "Code" section.

#### Parameters:

*the\_tokens* the array with tokens that describe the colour styling information, in particular the token with the default colour and the tokens with the colours of the special word categories

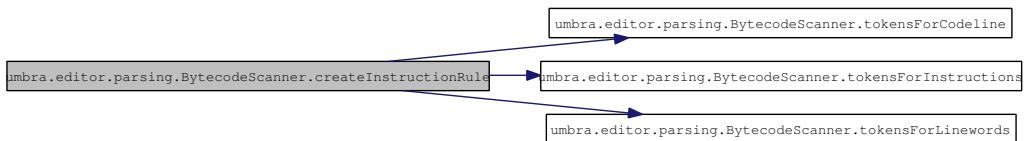
#### Returns:

the rule for colouring the words

References [umbra.editor.parsing.BytecodeScanner.tokensForCodeline\(\)](#), [umbra.editor.parsing.BytecodeScanner.tokensForInstructions\(\)](#), and [umbra.editor.parsing.BytecodeScanner.tokensForLinewords\(\)](#).

Referenced by [umbra.editor.parsing.BytecodeScanner.createRulesArray\(\)](#).

Here is the call graph for this function:



### 6.27.3.3 void umbra.editor.parsing.BytecodeScanner.tokensForCodeline (final IToken[ ] *the\_tokens*, final WordRule *the\_insrule*) [private]

This method associates in *the\_insrule* the words which occur in a line with the "Code" keyword with the token in *the\_tokens* under [ColorValues#SLOT\\_LINE](#).

#### Parameters:

*the\_tokens* the array with tokens, in particular with the [ColorValues#SLOT\\_LINE](#) token  
*the\_insrule* the rule in which the association is created

Referenced by [umbra.editor.parsing.BytecodeScanner.createInstructionRule\(\)](#).

### 6.27.3.4 void umbra.editor.parsing.BytecodeScanner.tokensForLinewords (final IToken[ ] *the\_tokens*, final WordRule *the\_insrule*) [private]

This method associates in *the\_insrule* the words which occur in a line with the "Line" keyword with the token in *the\_tokens* under [ColorValues#SLOT\\_LINE](#).

#### Parameters:

*the\_tokens* the array with tokens, in particular with the [ColorValues#SLOT\\_LINE](#) token

*the\_insrule* the rule in which the association is created

Referenced by `umbra.editor.parsing.BytecodeScanner.createInstructionRule()`.

#### **6.27.3.5 void `umbra.editor.parsing.BytecodeScanner.tokensForInstructions` (final `IToken[ ] the_tokens, final WordRule the_insrule)` [private]**

This method associates in `the_insrule` the words of the byte code `instructions` with the token in `the_tokens` under `ColorValues#SLOT_BTCTINSTR`.

**Parameters:**

*the\_tokens* the array with tokens, in particular with the `ColorValues#SLOT_BTCTINSTR` token  
*the\_insrule* the rule in which the association is created

Referenced by `umbra.editor.parsing.BytecodeScanner.createInstructionRule()`.

### **6.27.4 Member Data Documentation**

#### **6.27.4.1 final int `umbra.editor.parsing.BytecodeScanner.RULE_EOL = 0` [static, private]**

The number of the rule which is responsible for colour and text styling of the end-of-line comments.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

#### **6.27.4.2 final int `umbra.editor.parsing.BytecodeScanner.RULE_WORDS = 1` [static, private]**

The number of the rule which is responsible for colour and text styling of the words (as defined in `BytecodeWordDetector`).

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

#### **6.27.4.3 final int `umbra.editor.parsing.BytecodeScanner.RULE_INS_NUMBER = 2` [static, private]**

The number of the rule which is responsible for colour and text styling of the byte code instruction numbers at the beginning of a method line.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

#### **6.27.4.4 final int `umbra.editor.parsing.BytecodeScanner.RULE_HASH = 3` [static, private]**

The number of the rule which is responsible for colour and text styling of the numbers preceded by the hash (#) sign.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.5 final int umbra.editor.parsing.BytecodeScanner.RULE\_PERCENT = 4 [static, private]**

The number of the rule which is responsible for colour and text styling of the numbers preceded by the percent (%) sign.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.6 final int umbra.editor.parsing.BytecodeScanner.RULE\_PARENTHESSES = 5 [static, private]**

The number of the rule which is responsible for colour and text styling of the numbers enclosed in the parentheses ('(, ')').

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.7 final int umbra.editor.parsing.BytecodeScanner.RULE\_BRACES = 6 [static, private]**

The number of the rule which is responsible for colour and text styling of the line parts enclosed in the braces ('{, '}').

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.8 final int umbra.editor.parsing.BytecodeScanner.RULE\_NUMBER = 7 [static, private]**

The number of the rule which is responsible for colour and text styling of the numbers (without #, %, or surrounding parentheses).

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.9 final int umbra.editor.parsing.BytecodeScanner.RULE\_WHITESPACE = 8 [static, private]**

The number of the rule which is responsible for colour and text styling of the whitespace areas.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.10 final int umbra.editor.parsing.BytecodeScanner.RULE\_COMMENT = 9 [static, private]**

The number of the rule which is responsible for colour and text styling of comments.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.11 final int umbra.editor.parsing.BytecodeScanner.RULE\_ANNOT = 10 [static, private]**

The number of the rule which is responsible for colour and text styling of BML annotations areas ending with @\*\|.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.12 final int umbra.editor.parsing.BytecodeScanner.RULE\_ANNOT\_SIMPLE = 11**  
[static, private]

The number of the rule which is responsible for colour and text styling of BML annotations areas ending with \*/.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

**6.27.4.13 final int umbra.editor.parsing.BytecodeScanner.NUMBER\_OF\_RULES = RULE\_ANNOT\_SIMPLE + 1** [static, private]

The number of colouring rules used in this class.

Referenced by `umbra.editor.parsing.BytecodeScanner.createRulesArray()`.

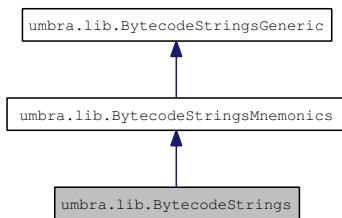
The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/[BytecodeScanner.java](#)

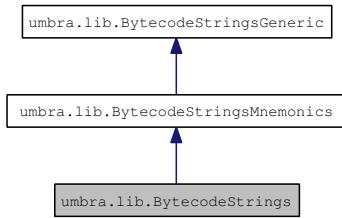
## 6.28 **umbra.lib.BytecodeStrings** Class Reference

The container for all the byte code and BML strings.

Inheritance diagram for **umbra.lib.BytecodeStrings**:



Collaboration diagram for **umbra.lib.BytecodeStrings**:



### Static Public Attributes

- static final String[ ] **BML\_KEYWORDS**

*This constant contains an array with all the BML keywords.*

- static final String[ ] **LINE\_KEYWORDS**

*This constant contains an array with all the keywords that occur in the line numbers area.*

- static final String[ ] **CODE\_KEYWORDS**

*This constant contains an array with all the keywords that occur in the Code area.*

- static final String[ ] **HEADER\_PREFIX**

*This constant contains an array with all the possible prefixes of method headers in byte code text files.*

- static final String[ ] **THROWS\_PREFIX**

*This constant contains an array with all the possible prefixes of throw lines in byte code text files.*

- static final String[ ] **PRIMITIVE\_TYPE\_NAMES**

*The names of base byte code types relevant for array [instructions](#).*

- static final char[ ] **WHITE SPACE\_CHARACTERS** = { ' ', '\t', '\n', '\r' }

*The array which contains all the characters we consider here to be whitespace characters.*

## Private Member Functions

- [BytecodeStrings \(\)](#)

*Private constructor added to prevent the creation of objects of this type.*

### 6.28.1 Detailed Description

The container for all the byte code and BML strings.

It inherits the byte code mnemonics as well as strings indicating starts and ends of comments and BML areas from [BytecodeStringsMnemonics](#). It contributes

- BML keywords (e.g. requires),
- BML expression keywords (e.g.

**Returns:**

) ,

- keywords for Line numbers section,
- keywords for Code section,
- keywords in method, classes, and package headers,
- keywords in throws section, and
- primitive types names.

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.28.2 Constructor & Destructor Documentation

#### 6.28.2.1 [umbra.lib.BytecodeStrings.BytecodeStrings \(\) \[private\]](#)

Private constructor added to prevent the creation of objects of this type.

### 6.28.3 Member Data Documentation

#### 6.28.3.1 [final String \[\] umbra.lib.BytecodeStrings.BML\\_KEYWORDS \[static\]](#)

**Initial value:**

```
new String[] {
    "invariant",
    "assert",
    "requires",
    "{ |",
    "| }",
    "precondition",
    "modifies",
    "ensures",
    "exsures",
    "\\result",
    "loop_specification",
    "modifies",
    "loop_inv",
    "decreases"}
```

This constant contains an array with all the BML keywords.

The BML lines are handled by [umbra.instructions.ast.AnnotationLineController](#) class.

FIXME: this should be retrieved from BMMLib; <https://mobius.ucd.ie/ticket/551>

#### **6.28.3.2 final String [ ] umbra.lib.BytocodeStrings.LINE\_KEYWORDS [static]**

**Initial value:**

```
new String[] {"Line", "numbers",
              "Local",
              "variable",
              "table"}
```

This constant contains an array with all the keywords that occur in the line numbers area.

This area is not fully handled yet.

FIXME: add the handling of this area; <https://mobius.ucd.ie/ticket/547>

#### **6.28.3.3 final String [ ] umbra.lib.BytocodeStrings.CODE\_KEYWORDS [static]**

**Initial value:**

```
new String[] {"Code",
              "max_stack",
              "max_locals",
              "code_length"}
```

This constant contains an array with all the keywords that occur in the Code area.

This area is not fully handled yet.

FIXME: add the handling of this area; <https://mobius.ucd.ie/ticket/548>

#### **6.28.3.4 final String [ ] umbra.lib.BytocodeStrings.HEADER\_PREFIX [static]**

**Initial value:**

```
new String[] {"public",
              "abstract",
```

```
"static", "void",
"private",
"int", "char",
"protected",
"boolean",
"String", "byte",
"package",
"class", "}" }
```

This constant contains an array with all the possible prefixes of method headers in byte code text files.

The header lines are handled by [umbra.instructions.ast.HeaderLineController](#) class.

#### **6.28.3.5 final String [ ] umbra.lib.BytecodeStrings.THROWS\_PREFIX [static]**

**Initial value:**

```
new String[] {"throws",
              "Exception",
              "From"}
```

This constant contains an array with all the possible prefixes of throw lines in byte code text files.

The throw lines are handled by [umbra.instructions.ast.ThrowsLineController](#) class.

FIXME: add the handling of this area; <https://mobius.ucd.ie/ticket/549>

#### **6.28.3.6 final String [ ] umbra.lib.BytecodeStrings.PRIMITIVE\_TYPE\_NAMES [static]**

**Initial value:**

```
{"boolean", "char",
   "float", "double",
   "byte", "short",
   "int", "long"}
```

The names of base byte code types relevant for array [instructions](#).

These are the primitive types.

#### **6.28.3.7 final char [ ] umbra.lib.BytecodeStrings.WHITESPACE\_CHARACTERS = {' ', '\t', '\n', '\r' } [static]**

The array which contains all the characters we consider here to be whitespace characters.

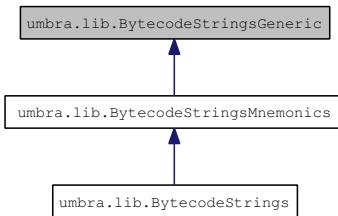
The documentation for this class was generated from the following file:

- source/umbra/lib/[BytecodeStrings.java](#)

## 6.29 umbra.lib.BytecodeStringsGeneric Class Reference

The class is a container for all byte code [instructions](#) and the sequences that indicate the starts and ends of comments or BML annotation areas.

Inheritance diagram for umbra.lib.BytecodeStringsGeneric:



### Static Public Attributes

- static final String[] [INSTRUCTIONS](#)  
*All the byte code mnemonics.*
- static final String [COMMENT\\_LINE\\_START](#) = "/\*"  
*This string contains the multi-line comment start.*
- static final String [COMMENT\\_LINE\\_END](#) = "\*/"  
*This string contains the multi-line comment end.*
- static final String [SINGLE\\_LINE\\_COMMENT\\_MARK](#) = "//"  
*The string which starts a single line comment in a byte code file.*
- static final int [SINGLE\\_LINE\\_COMMENT\\_MARK\\_LEN](#)  
*The length of the single line comment marker.*
- static final String [ANNOT\\_LINE\\_START](#) = "/\*@"  
*This string contains the BML annotation comment start.*
- static final String [ANNOT\\_LINE\\_END](#) = "@\*/"  
*This string contains the BML annotation comment end i.e.*
- static final String [ANNOT\\_LINE\\_END\\_SIMPLE](#) = "\*/"  
*This string contains the BML annotation comment end.*

### Protected Member Functions

- [BytecodeStringsGeneric \(\)](#)  
*Private constructor added to prevent the creation of objects of this type.*

### 6.29.1 Detailed Description

The class is a container for all byte code [instructions](#) and the sequences that indicate the starts and ends of comments or BML annotation areas.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.29.2 Constructor & Destructor Documentation

#### 6.29.2.1 `umbra.lib.BytecodeStringsGeneric.BytecodeStringsGeneric()` [protected]

Private constructor added to prevent the creation of objects of this type.

### 6.29.3 Member Data Documentation

#### 6.29.3.1 `final String [] umbra.lib.BytecodeStringsGeneric.INSTRUCTIONS` [static]

All the byte code mnemonics.

#### 6.29.3.2 `final String umbra.lib.BytecodeStringsGeneric.COMMENT_LINE_START = "/*"` [static]

This string contains the multi-line comment start.

#### 6.29.3.3 `final String umbra.lib.BytecodeStringsGeneric.COMMENT_LINE_END = "*/"` [static]

This string contains the multi-line comment end.

#### 6.29.3.4 `final String umbra.lib.BytecodeStringsGeneric.SINGLE_LINE_COMMENT_MARK = "://"` [static]

The string which starts a single line comment in a byte code file.

#### 6.29.3.5 `final int umbra.lib.BytecodeStringsGeneric.SINGLE_LINE_COMMENT_MARK_LEN` [static]

**Initial value:**

`SINGLE_LINE_COMMENT_MARK.length()`

The length of the single line comment marker.

**6.29.3.6 final String umbra.lib.BytecodeStringsGeneric.ANOT\_LINE\_START = "/\*@"**  
[static]

This string contains the BML annotation comment start.

**6.29.3.7 final String umbra.lib.BytecodeStringsGeneric.ANOT\_LINE\_END = "@\*/"**  
[static]

This string contains the BML annotation comment end i.e.

@\*\/.

**6.29.3.8 final String umbra.lib.BytecodeStringsGeneric.ANOT\_LINE\_END\_SIMPLE = "\*/"**  
[static]

This string contains the BML annotation comment end.

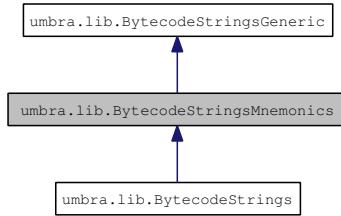
The documentation for this class was generated from the following file:

- source/umbra/lib/[BytecodeStringsGeneric.java](#)

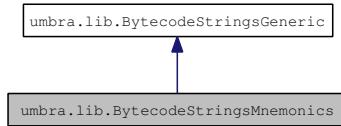
## 6.30 umbra.lib.BytecodeStringsMnemonics Class Reference

The container for all the byte code mnemonic strings.

Inheritance diagram for umbra.lib.BytecodeStringsMnemonics:



Collaboration diagram for umbra.lib.BytecodeStringsMnemonics:



### Static Public Attributes

- static final String[] ARITHMETIC\_INS

*This constant contains an array with all the names of instructions handled in [umbra.instructions.ast.ArithmeticInstruction](#) class.*

- static final String[] ICONST\_INS

*This constant contains an array with all the names of instructions handled in [umbra.instructions.ast.IConstInstruction](#) class.*

- static final String[] LOAD\_STORE\_INS

*This constant contains an array with all the names of instructions handled in [umbra.instructions.ast.LoadStoreConstInstruction](#) class.*

- static final String[] LOAD\_STORE\_ARRAY\_INS

*This constant contains an array with all the names of instructions handled in [umbra.instructions.ast.LoadStoreArrayInstruction](#) class.*

- static final String[] SINGLE\_INS

*This constant contains an array with all the names of instructions handled in [umbra.instructions.ast.SingleInstruction](#) class.*

- static final String[] PUSH\_INS = new String[] {"bipush", "sipush"}

*This constant contains an array with all the names of instructions handled in [umbra.instructions.ast.PushInstruction](#) class.*

- static final String[] JUMP\_INS

*This constant contains an array with all the names of instructions handled in [umbra.instructions.ast.JumpInstruction](#) class.*

- static final String[ ] **CONV\_INS**

*This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.ConversionInstruction](#) class.*

- static final String[ ] **IINC\_INS** = new String[ ] {"iinc"}

*This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.IincInstruction](#) class.*

- static final String[ ] **STACK\_INS**

*This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.StackInstruction](#) class.*

- static final String[ ] **ARRAY\_INS** = new String[ ] {"newarray"}

*This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.ArrayInstruction](#) class.*

- static final String[ ] **NEW\_INS**

*This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.NewInstruction](#) class.*

- static final String[ ] **FIELD\_INS**

*This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.FieldInstruction](#) class.*

- static final String[ ] **INVOKE\_INS**

*This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.InvokeInstruction](#) class.*

- static final int **INVOKEINTERFACE\_NO** = 0

*Contains the index to [INVOKE\\_INS](#) of "invokeinterface".*

- static final String[ ] **LDC\_INS**

*This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.LdcInstruction](#) class.*

- static final String[ ] **UNCLASSIFIED\_INS**

*This array contains [instructions](#) which are not handled by the Umbra plugin.*

## Protected Member Functions

- [BytocodeStringsMnemonics \(\)](#)

*Private constructor added to prevent the creation of objects of this type.*

### 6.30.1 Detailed Description

The container for all the byte code mnemonic strings.

Except from a flat list of mnemonics it contains arrays of mnemonics divided to different classes which are represented in [umbra.instructions.ast](#) package. It inherits the flat list of all [instructions](#) from [BytecodeStringsGeneric](#).

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.30.2 Constructor & Destructor Documentation

#### 6.30.2.1 [umbra.lib.BytecodeStringsMnemonics.BytecodeStringsMnemonics \(\)](#) [protected]

Private constructor added to prevent the creation of objects of this type.

### 6.30.3 Member Data Documentation

#### 6.30.3.1 [final String \[\] umbra.lib.BytecodeStringsMnemonics.ARITHMETIC\\_INS](#) [static]

**Initial value:**

```
new String[] {"dadd", "ddiv",
              "dmul", "dneg",
              "drem", "dsub",
              "fadd", "fdiv",
              "fmul", "fneg",
              "frem", "fsub",
              "iadd", "iand",
              "imul", "idiv",
              "ineg", "ior",
              "isub", "irem",
              "ishl",
              "iushr", "ixor",
              "lsub", "ladd",
              "land", "ldiv",
              "lmul", "lneg",
              "lor", "lrem",
              "lshl", "lshr",
              "lushr", "lxor"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.ArithmeticInstruction](#) class.

#### 6.30.3.2 [final String \[\] umbra.lib.BytecodeStringsMnemonics.ICONST\\_INS](#) [static]

**Initial value:**

```
new String[] {"iconst_0",
              "iconst_1",
              "iconst_2",
              "iconst_3",
              "iconst_4",
              "iconst_5"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.IConstInstruction](#) class.

#### **6.30.3.3 final String [ ] umbra.lib.BytecodeStringsMnemonics.LOAD\_STORE\_INS [static]**

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.LoadStoreConstInstruction](#) class.

#### **6.30.3.4 final String [ ] umbra.lib.BytecodeStringsMnemonics.LOAD\_STORE\_ARRAY\_INS [static]**

**Initial value:**

```
{"aaload",
     "aastore",
     "baload",
     "bastore",
     "caload",
     "castore",
     "daload",
     "dastore",
     "faload",
     "fastore",
     "iaload",
     "iastore",
     "laload",
     "lastore",
     "saload",
     "sastore"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.LoadStoreArrayInstruction](#) class.

#### **6.30.3.5 final String [ ] umbra.lib.BytecodeStringsMnemonics.SINGLE\_INS [static]**

**Initial value:**

```
new String[] {"acconst_null",
               "arraylength",
               "athrow", "lcmp",
               "monitorenter",
               "monitorexit",
               "areturn", "dreturn",
               "freturn", "ireturn",
               "lreturn", "return",
               "dup", "dup_x1",
               "dup_x2", "dup2",
               "dup2_x1", "dup2_x2",
               "pop", "pop2",
               "swap"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.SingleInstruction](#) class.

---

**6.30.3.6 final String [ ] umbra.lib.BytecodeStringsMnemonics.PUSH\_INS = new String[ ] {"bipush", "sipush"} [static]**

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.PushInstruction](#) class.

**6.30.3.7 final String [ ] umbra.lib.BytecodeStringsMnemonics.JUMP\_INS [static]**

**Initial value:**

```
new String[] {"goto", "goto_w",
              "if_acmpeq",
              "if_acmpne",
              "if_icmpeq",
              "if_icmpge",
              "if_icmpgt",
              "if_icmple",
              "if_icmplt",
              "if_icmpne", "ifeq",
              "ifge", "ifgt", "ifle",
              "iflt", "ifne",
              "ifnonnull", "ifnull",
              "jsr", "jsr_w",
              "lookupswitch",
              "tableswitch"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.JumpInstruction](#) class.

**6.30.3.8 final String [ ] umbra.lib.BytecodeStringsMnemonics.CONV\_INS [static]**

**Initial value:**

```
new String[] {"d2f",
              "d2i",
              "d2l",
              "f2d",
              "f2i",
              "f2l",
              "i2b",
              "i2c",
              "i2d",
              "i2f",
              "i2l",
              "i2s",
              "l2d",
              "l2f",
              "l2i"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.ConversionInstruction](#) class.

**6.30.3.9 final String [ ] umbra.lib.BytecodeStringsMnemonics.IINC\_INS = new String[ ] {"iinc"} [static]**

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.IincInstruction](#) class.

**6.30.3.10 final String [ ] umbra.lib.BytecodeStringsMnemonics.STACK\_INS [static]****Initial value:**

```
new String[] {"aload", "astore",
              "dload", "dstore",
              "fload", "fstore",
              "iload", "istore",
              "lload", "lstore"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.StackInstruction](#) class.

**6.30.3.11 final String [ ] umbra.lib.BytecodeStringsMnemonics.ARRAY\_INS = new String[ ]
{ "newarray" } [static]**

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.ArrayInstruction](#) class.

**6.30.3.12 final String [ ] umbra.lib.BytecodeStringsMnemonics.NEW\_INS [static]****Initial value:**

```
new String[] {"anewarray",
              "checkcast",
              "instanceof", "new"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.NewInstruction](#) class.

**6.30.3.13 final String [ ] umbra.lib.BytecodeStringsMnemonics.FIELD\_INS [static]****Initial value:**

```
new String[] {"getfield",
              "getstatic",
              "putfield",
              "putstatic"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.FieldInstruction](#) class.

**6.30.3.14 final String [ ] umbra.lib.BytecodeStringsMnemonics.INVOKE\_INS [static]****Initial value:**

```
new String[] {"invokeinterface",
              "invokespecial",
              "invokestatic",
              "invokevirtual"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.InvokeInstruction](#) class.

**6.30.3.15 final int umbra.lib.BytecodeStringsMnemonics.INVOKEINTERFACE\_NO = 0  
[static]**

Contains the index to [INVOKE\\_INS](#) of "invokeinterface".

**6.30.3.16 final String [] umbra.lib.BytecodeStringsMnemonics.LDC\_INS [static]**

**Initial value:**

```
new String[] {"ldc", "ldc_w",
              "ldc2_w"}
```

This constant contains an array with all the names of [instructions](#) handled in [umbra.instructions.ast.LdcInstruction](#) class.

**6.30.3.17 final String [] umbra.lib.BytecodeStringsMnemonics.UNCLASSIFIED\_INS  
[static]**

**Initial value:**

```
new String[] {"breakpoint",
              "multilinearray",
              "dcmpg",
              "dcmpl",
              "dconst",
              "fcmpg",
              "fcmpl",
              "fconst",
              "iconst",
              "imdep1",
              "imdep2",
              "lconst",
              "nop", "ret"}
```

This array contains [instructions](#) which are not handled by the Umbra plugin.

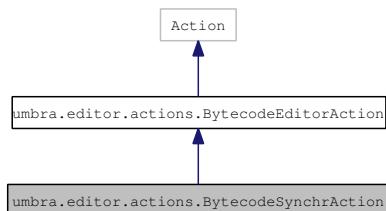
The documentation for this class was generated from the following file:

- source/umbra/lib/[BytecodeStringsMnemonics.java](#)

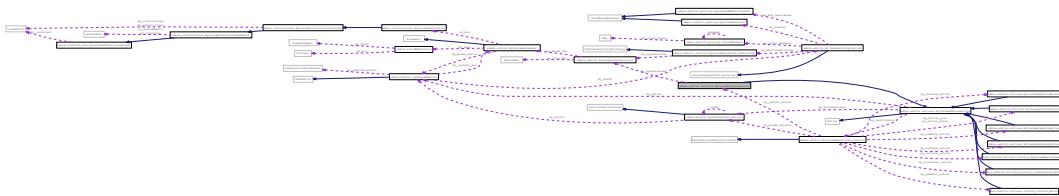
## 6.31 umbra.editor.actions.BytecodeSynchrAction Class Reference

This class defines action of the synchronisation for a byte code position with an appropriate point in the source code.

Inheritance diagram for umbra.editor.actions.BytecodeSynchrAction:



Collaboration diagram for umbra.editor.actions.BytecodeSynchrAction:



### Public Member Functions

- **BytecodeSynchrAction** (final BytecodeEditorContributor a\_contributor, final BytecodeContribution a\_bytecode\_contribution)
 

*The constructor of the action.*
- final void **run** ()
 

*This method runs the synchronisation of the current byte code with the source code.*
- void **synchronizeBS** (final int a\_pos) throws UmbraLocationException, UmbraSynchronisationException
 

*Highlights the area in the source code `editor` which corresponds to the marked area in the byte code `editor`.*

### Static Public Member Functions

- static void **wrongSynchronisationMessage** (final Shell a\_shell, final String a\_title)
 

*Displays the message that no source code instruction can be reasonably associated with the given position.*

### Private Member Functions

- **DocumentSynchroniser getDocSynch** ()
 

*This method lazily provides the object which performs the synchronisation operations.*

## Private Attributes

- [DocumentSynchroniser my\\_synchroniser](#)

*This is an object which handles the calculations of the synchronisation positions.*

### 6.31.1 Detailed Description

This class defines action of the synchronisation for a byte code position with an appropriate point in the source code.

**See also:**

[umbra.editor.BytecodeDocument](#)

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

**Version:**

a-01

### 6.31.2 Constructor & Destructor Documentation

#### 6.31.2.1 [umbra.editor.actions.BytecodeSynchrAction](#).BytecodeSynchrAction (final BytecodeEditorContributor *a\_contributor*, final BytecodeContribution *a\_bytecode\_contribution*)

The constructor of the action.

It only registers the name of the action in the eclipse environment.

**Parameters:**

*a\_contributor* the manager that initialises all the [actions](#) within the byte code plugin

*a\_bytecode\_contribution* the GUI elements contributed to the eclipse GUI by the byte code [editor](#).  
This reference should be the same as in the parameter *a\_contributor*.

### 6.31.3 Member Function Documentation

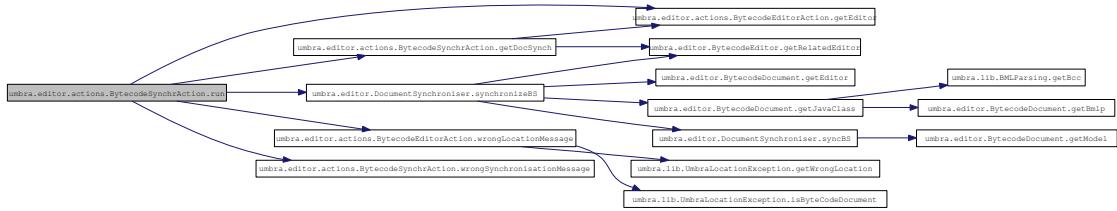
#### 6.31.3.1 [final void umbra.editor.actions.BytecodeSynchrAction.run \(\)](#)

This method runs the synchronisation of the current byte code with the source code.

It retrieves the current selection, extracts the offset of the beginning of the selection and shows the related Java source code document that corresponds to the offset.

References [umbra.editor.actions.BytecodeSynchrAction.getDocSynch\(\)](#), [umbra.editor.actions.BytecodeEditorAction.getEditor\(\)](#), [umbra.editor.DocumentSynchroniser.synchronizeBS\(\)](#), [umbra.editor.actions.BytecodeEditorAction.wrongLocationMessage\(\)](#), [and](#) [umbra.editor.actions.BytecodeSynchrAction.wrongSynchronisationMessage\(\)](#).

Here is the call graph for this function:



### 6.31.3.2 static void umbra.editor.actions.BytecodeSynchrAction.wrongSynchronisationMessage (final Shell *a\_shell*, final String *a\_title*) [static]

Displays the message that no source code instruction can be reasonably associated with the given position.

#### Parameters:

- a\_shell* the shell which displays the message
- a\_title* the title of the message window

Referenced by `umbra.editor.actions.BytecodeSynchrAction.run()`.

### 6.31.3.3 DocumentSynchroniser umbra.editor.actions.BytecodeSynchrAction.getDocSynch () [private]

This method lazily provides the object which performs the synchronisation operations.

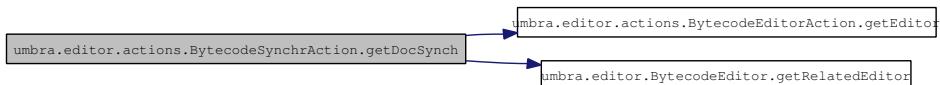
#### Returns:

- a `DocumentSynchroniser` which performs the synchronisation operations

References `umbra.editor.actions.BytecodeEditorAction.getEditor()`, `umbra.editor.BytecodeEditor.getRelatedEditor()`, and `umbra.editor.actions.BytecodeSynchrAction.my_synchroniser`.

Referenced by `umbra.editor.actions.BytecodeSynchrAction.run()`, and `umbra.editor.actions.BytecodeSynchrAction.synchronizeBS()`.

Here is the call graph for this function:



### 6.31.3.4 void umbra.editor.actions.BytecodeSynchrAction.synchronizeBS (final int *a\_pos*) throws UmbraLocationException, UmbraSynchronisationException

Highlights the area in the source code `editor` which corresponds to the marked area in the byte code `editor`. Works correctly only inside a method body.

**Parameters:**

*a\_pos* index of line in byte code [editor](#). Lines in related source code [editor](#) corresponding to this line will be highlighted.

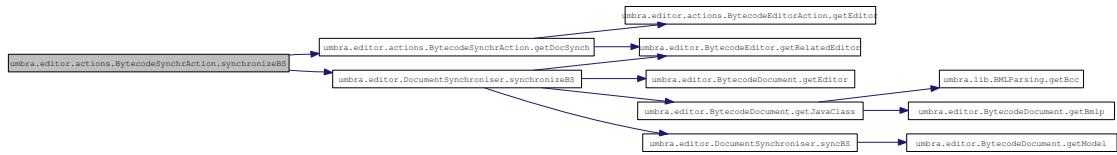
**Exceptions:**

*UmbraLocationException* in case a position is reached in the source code or byte code [editor](#) which does not exists there

*UmbraSynchronisationException* in case there is no instruction line which can be reasonably associated with the given position

References      [umbra.editor.actions.BytecodeSynchrAction.getDocSynch\(\)](#),      and      [umbra.editor.DocumentSynchroniser.synchronizeBS\(\)](#).

Here is the call graph for this function:



## 6.31.4 Member Data Documentation

### 6.31.4.1 DocumentSynchroniser [umbra.editor.actions.BytecodeSynchrAction.my\\_synchroniser](#) [private]

This is an object which handles the calculations of the synchronisation positions.

Referenced by [umbra.editor.actions.BytecodeSynchrAction.getDocSynch\(\)](#).

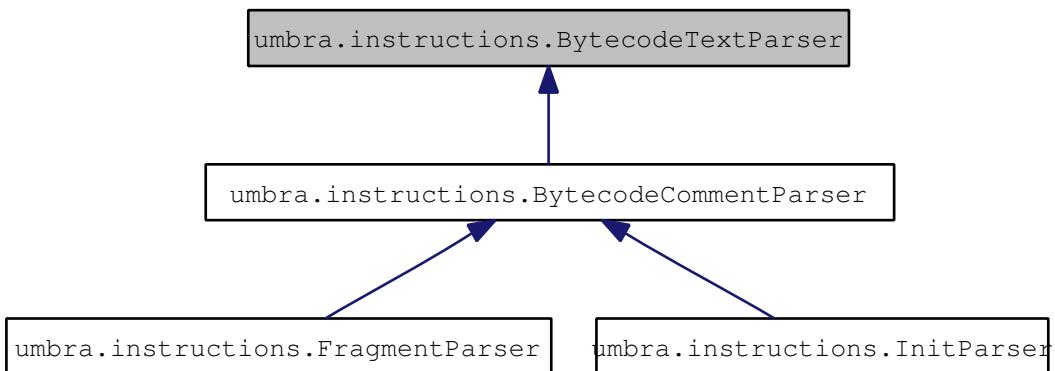
The documentation for this class was generated from the following file:

- [source/umbra/editor/actions/BytecodeSynchrAction.java](#)

## 6.32 `umbra.instructions.BytecodeTextParser` Class Reference

This class handles the operations which are common to all the document parsers that are used in Umbra.

Inheritance diagram for `umbra.instructions.BytecodeTextParser`:



Collaboration diagram for `umbra.instructions.BytecodeTextParser`:



### Public Member Functions

- `LinkedList getEditorLines ()`  
*Returns the list of all the lines in the internal representation.*
- `void addEditorLine (final int a_pos, final BytecodeLineController a_line)`  
*This method adds the specified line controller at the specified position.*
- `void addEditorLine (final BytecodeLineController a_line)`  
*This method appends the specified line controller at the end of the lines structure.*
- `LinkedList getInstructions ()`  
*Returns the list of all the lines with `instructions` in the internal representation.*

### Static Public Member Functions

- `static String extractCommentFromLine (final String a_line_text, final LineContext a_ctxt)`  
*The method checks if the given line contains a single line comment and extracts the comment string.*
- `static final String removeCommentFromLine (final String a_line)`

*Removes an one-line comment from a line of byte code.*

## Protected Member Functions

- **BytecodeTextParser ()**  
*This constructor initialises internal structure to represent `editor` lines and `instructions`.*
- abstract int **getPosOfLine** (final int a\_lineno)  
*Returns the position of the first character in the line of the given number.*
- abstract void **insertAt** (int a\_pos, String a\_string)  
*Inserts the given string in the current representation of the combined text (class+comments) at the indicated position.*
- abstract void **enrichWithComment** (final BytecodeLineController a\_line, final int a\_pos, final int a\_instno)  
*This method adds to the combination of the currently parsed text and the information from the comment structures the comment associated with the given line.*
- abstract void **enrichWithComment** (final BytecodeLineController a\_line, final int a\_instno)  
*This method adds to the combination of the currently parsed text and the information from the comment structures the text of the given instruction together with the comment associated with the line.*
- int **addInstruction** (final InstructionLineController a\_lc)  
*Adds the given instruction line controller to the collection of the instruction lines.*
- abstract void **adjustCommentsForInstruction** (final InstructionLineController a\_lc, final int an\_instrno)  
*The method updates the comments structures.*
- void **incInstructionNo ()**  
*Increases by one the current instruction number.*
- void **initInstructionNo ()**  
*Initialises the instruction counter to the first value.*
- void **updateAnnotations** (final LineContext a\_ctxt)  
*Assigns the method number included in the given line context to the annotation lines block at the end of the current collection of the `editor` lines.*
- int **getInstructionNoForLine** (final int a\_lineno)  
*Converts the given line number to the corresponding instruction number.*

## Static Protected Member Functions

- static MethodGen **getMethodGenFromDoc** (final BytecodeDocument a\_doc, final int a\_method\_no)  
throws UmbraMethodException  
*This method retrieves from the given byte code document the BCEL structure corresponding to the method of the given number.*

## Private Attributes

- LinkedList `my_editor_lines`  
*The list of all the lines in the current byte code `editor`.*
- int `my_instruction_no`  
*A temporary counter of instruction lines.*
- LinkedList `my_instructions`  
*The list of all the lines in the `editor` which contain codes of `instructions`.*

### 6.32.1 Detailed Description

This class handles the operations which are common to all the document parsers that are used in Umbra.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.32.2 Constructor & Destructor Documentation

#### 6.32.2.1 umbra.instructions.BytecodeTextParser() [protected]

This constructor initialises internal structure to represent `editor` lines and `instructions`.

References umbra.instructions.BytecodeTextParser.my\_editor\_lines, and  
umbra.instructions.BytecodeTextParser.my\_instructions.

### 6.32.3 Member Function Documentation

#### 6.32.3.1 static String umbra.instructions.BytecodeTextParser.extractCommentFromLine (final String *a\_line\_text*, final LineContext *a\_ctxt*) [static]

The method checks if the given line contains a single line comment and extracts the comment string.

In case there is no comment in the line, it returns `null`. In case the parsing context is such that we are inside a many-line comment, then the comment inside a line is always empty. Additionally, this method removes the end-of-line char from the string.

#### Parameters:

*a\_line\_text* the line to check for my\_eolcomments  
*a\_ctxt* the parsing context for the line

#### Returns:

comment or `null` in case there is no comment in the line

References `umbra.instructions.LineContext.isInsideComment()`.

Referenced by `umbra.instructions.BytecodeCommentParser.getLineFromDoc()`.

Here is the call graph for this function:



### **6.32.3.2 static MethodGen umbra.instructions.BytecodeTextParser.getMethodGenFromDoc (final BytecodeDocument a\_doc, final int a\_method\_no) throws UmbraMethodException [static, protected]**

This method retrieves from the given byte code document the BCEL structure corresponding to the method of the given number.

This method checks if there are enough methods in the BCEL structure of the document and in case there are not enough of them it throws an exception.

**Parameters:**

- a\_doc* a document to retrieve the BCEL structure of a method
- a\_method\_no* the method number of the method to retrieve the structure for

**Returns:**

the BCEL structure which describes the method

**Exceptions:**

*UmbraMethodException* in case the given method number is wrong

Referenced by `umbra.instructions.InitParser.swallowMethod()`.

### **6.32.3.3 static final String umbra.instructions.BytecodeTextParser.removeCommentFromLine (final String a\_line) [static]**

Removes an one-line comment from a line of byte code.

**Parameters:**

- a\_line* a line of byte code

**Returns:**

the byte code line without end-of-line comment and final whitespace

Referenced by `umbra.instructions.BytecodeCommentParser.getLineFromDoc()`.

### **6.32.3.4 LinkedList umbra.instructions.BytecodeTextParser.getEditorLines ()**

Returns the list of all the lines in the internal representation.

This method may only be called once to export fully generated list of lines.

**Returns:**

the list of the [BytecodeLineController](#) objects that represent all the lines in the currently parsed document

References [umbra.instructions.BytecodeTextParser.my\\_editor\\_lines](#).

Referenced by [umbra.instructions.BytecodeControllerContainer.init\(\)](#).

### 6.32.3.5 void [umbra.instructions.BytecodeTextParser.addEditorLine \(final int a\\_pos, final BytecodeLineController a\\_line\)](#)

This method adds the specified line controller at the specified position.

It shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

**Parameters:**

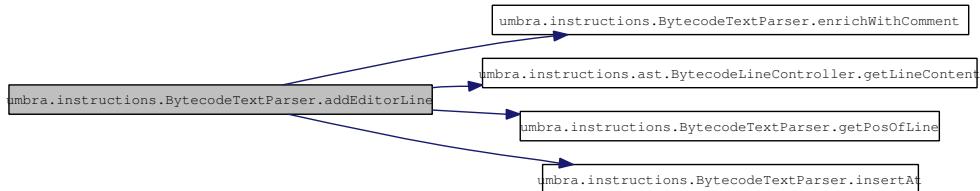
*a\_pos* the position in the document where to insert the line

*a\_line* the line to be inserted

References [umbra.instructions.BytecodeTextParser.enrichWithComment\(\)](#), [umbra.instructions.ast.BytecodeLineController.getLineContent\(\)](#), [umbra.instructions.BytecodeTextParser.getPosOfLine\(\)](#), [umbra.instructions.BytecodeTextParser.insertAt\(\)](#), [umbra.instructions.BytecodeTextParser.my\\_editor\\_lines](#), and [umbra.instructions.BytecodeTextParser.my\\_instruction\\_no](#).

Referenced by [umbra.instructions.InitParser.swallowClassHeader\(\)](#), [umbra.instructions.BytecodeCommentParser.swallowEmptyLines\(\)](#), [umbra.instructions.InitParser.swallowMethod\(\)](#), and [umbra.instructions.InitParser.swallowMethodHeader\(\)](#).

Here is the call graph for this function:



### 6.32.3.6 abstract int [umbra.instructions.BytecodeTextParser.getPosOfLine \(final int a\\_lineno\)](#) [protected, pure virtual]

Returns the position of the first character in the line of the given number.

**Parameters:**

*a\_lineno* the number of the line to find the position for

**Returns:**

the position of the first character in the line

Implemented in [umbra.instructions.BytecodeCommentParser](#).

Referenced by [umbra.instructions.BytecodeTextParser.addEditorLine\(\)](#).

---

**6.32.3.7 abstract void umbra.instructions.BytecodeTextParser.insertAt (int *a\_pos*, String *a\_string*) [protected, pure virtual]**

Inserts the given string in the current representation of the combined text (class+comments) at the indicated position.

**Parameters:**

- a\_pos* the position to insert the string at
- a\_string* the string to insert

Referenced by [umbra.instructions.BytecodeTextParser.addEditorLine\(\)](#).

**6.32.3.8 abstract void umbra.instructions.BytecodeTextParser.enrichWithComment (final BytecodeLineController *a\_line*, final int *a\_pos*, final int *a\_instno*) [protected, pure virtual]**

This method adds to the combination of the currently parsed text and the information from the comment structures the comment associated with the given line.

If the given line controller is not an [InstructionLineController](#) then the method does nothing.

**Parameters:**

- a\_line* a line controller to associate comments with
- a\_pos* the position in the combined text where the comment is to be added
- a\_instno* the number of a instruction with which the comment should be associated

Implemented in [umbra.instructions.BytecodeCommentParser](#).

Referenced by [umbra.instructions.BytecodeTextParser.addEditorLine\(\)](#).

**6.32.3.9 abstract void umbra.instructions.BytecodeTextParser.enrichWithComment (final BytecodeLineController *a\_line*, final int *a\_instno*) [protected, pure virtual]**

This method adds to the combination of the currently parsed text and the information from the comment structures the text of the given instruction together with the comment associated with the line.

We assume that the text of the line controller is not already in the combined text string. If the given line controller is not an [InstructionLineController](#) then the method only appends the content of the given line controller

**Parameters:**

- a\_line* a line controller to associate comments with
- a\_instno* the number of a instruction with which the comment should be associated

Implemented in [umbra.instructions.BytecodeCommentParser](#).

**6.32.3.10 void umbra.instructions.BytecodeTextParser.addEditorLine (final BytecodeLineController *a\_line*)**

This method appends the specified line controller at the end of the lines structure.

**Parameters:**

*a\_line* the line to be inserted

References [umbra.instructions.BytecodeTextParser.enrichWithComment\(\)](#), [umbra.instructions.BytecodeTextParser.my\\_editor\\_lines](#), and [umbra.instructions.BytecodeTextParser.my\\_instruction\\_no](#).

Here is the call graph for this function:

**6.32.3.11 LinkedList umbra.instructions.BytecodeTextParser.getInstructions ()**

Returns the list of all the lines with [instructions](#) in the internal representation.

This method may only be called once to export fully generate list of lines.

**Returns:**

the list of the [BytecodeLineController](#) objects that represent the lines with [instructions](#) in the currently parsed document

References [umbra.instructions.BytecodeTextParser.my\\_instructions](#).

Referenced by [umbra.instructions.BytecodeControllerContainer.init\(\)](#).

**6.32.3.12 int umbra.instructions.BytecodeTextParser.addInstruction (final InstructionLineController *a\_lc*) [protected]**

Adds the given instruction line controller to the collection of the instruction lines.

Additionally, this method handles the adding of the comments from the currently parsed document to the structures which represent the comments internally. It is done here as all the comments are associated with the instruction lines.

**Parameters:**

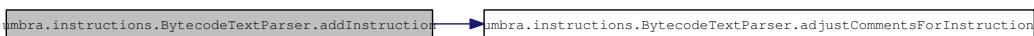
*a\_lc* the line controller to add

**Returns:**

the number of the currently added instruction

References [umbra.instructions.BytecodeTextParser.adjustCommentsForInstruction\(\)](#), [umbra.instructions.BytecodeTextParser.my\\_instruction\\_no](#), and [umbra.instructions.BytecodeTextParser.my\\_instructions](#).

Here is the call graph for this function:



**6.32.3.13 abstract void umbra.instructions.BytecodeTextParser.adjustCommentsForInstruction  
(final InstructionLineController *a\_lc*, final int *an\_instrno*) [protected, pure  
virtual]**

The method updates the comments structures.

**Parameters:**

*a\_lc* the line controller to associate the comments with  
*an\_instrno* the instruction number of the given controller

Implemented in [umbra.instructions.BytecodeCommentParser](#).

Referenced by [umbra.instructions.BytecodeTextParser.addInstruction\(\)](#).

**6.32.3.14 void umbra.instructions.BytecodeTextParser.incInstructionNo () [protected]**

Increases by one the current instruction number.

References [umbra.instructions.BytecodeTextParser.my\\_instruction\\_no](#).

**6.32.3.15 void umbra.instructions.BytecodeTextParser.initInstructionNo () [protected]**

Initialises the instruction counter to the first value.

References [umbra.instructions.BytecodeTextParser.my\\_instruction\\_no](#).

Referenced by [umbra.instructions.InitParser.runParsing\(\)](#).

**6.32.3.16 void umbra.instructions.BytecodeTextParser.updateAnnotations (final LineContext  
*a\_ctxt*) [protected]**

Assigns the method number included in the given line context to the annotation lines block at the end of the current collection of the [editor](#) lines.

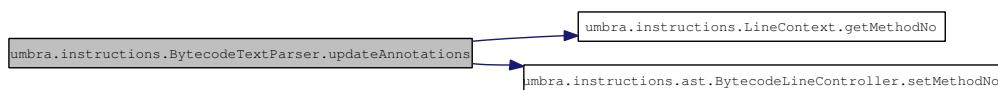
**Parameters:**

*a\_ctxt* a context with the method number to assign

References [umbra.instructions.LineContext.getMethodNo\(\)](#), [umbra.instructions.BytecodeTextParser.my\\_editor\\_lines](#), and [umbra.instructions.ast.BytecodeLineController.setMethodNo\(\)](#).

Referenced by [umbra.instructions.InitParser.runParsing\(\)](#).

Here is the call graph for this function:



### 6.32.3.17 int umbra.instructions.BytecodeTextParser.getInstructionNoForLine (final int *a\_lineno*) [protected]

Converts the given line number to the corresponding instruction number.

This method returns the instruction number only for lines that contain [instructions](#). For other lines the method returns -1.

**Parameters:**

*a\_lineno* the line number for which the instruction number is retrieved

**Returns:**

the number of instruction or -1 in case the given line number does not contain an instruction

References                   umbra.instructions.BytecodeTextParser.my\_editor\_lines,                   and  
umbra.instructions.BytecodeTextParser.my\_instructions.

## 6.32.4 Member Data Documentation

### 6.32.4.1 LinkedList umbra.instructions.BytecodeTextParser.my\_editor\_lines [private]

The list of all the lines in the current byte code [editor](#).

These lines are stored as objects the classes of which are subclasses of [BytecodeLineController](#).

Referenced           by           umbra.instructions.BytecodeTextParser.addEditorLine(),           um-  
bra.instructions.BytecodeTextParser.BytecodeTextParser(), umbra.instructions.BytecodeTextParser.getEditorLines(),  
umbra.instructions.BytecodeTextParser.getInstructionNoForLine(),                   and           um-  
bra.instructions.BytecodeTextParser.updateAnnotations().

### 6.32.4.2 int umbra.instructions.BytecodeTextParser.my\_instruction\_no [private]

A temporary counter of instruction lines.

It is used to synchronise the currently parsed document with an old comments structure. This number is a sequence number increased by one with each instruction (not the byte code label number).

Referenced           by           umbra.instructions.BytecodeTextParser.addEditorLine(),           um-  
bra.instructions.BytecodeTextParser.addInstruction(), umbra.instructions.BytecodeTextParser.incInstructionNo(),  
and umbra.instructions.BytecodeTextParser.initInstructionNo().

### 6.32.4.3 LinkedList umbra.instructions.BytecodeTextParser.my\_instructions [private]

The list of all the lines in the [editor](#) which contain codes of [instructions](#).

These are represented as objects the classes of which are subclasses of [InstructionLineController](#).

Referenced           by           umbra.instructions.BytecodeTextParser.addInstruction(),           um-  
bra.instructions.BytecodeTextParser.BytecodeTextParser(), umbra.instructions.BytecodeTextParser.getInstructionNoForLine(),  
and umbra.instructions.BytecodeTextParser.getInstructions().

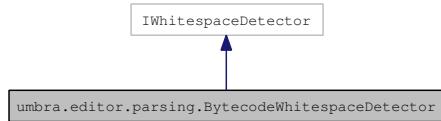
The documentation for this class was generated from the following file:

- source/umbra/instructions/[BytecodeTextParser.java](#)

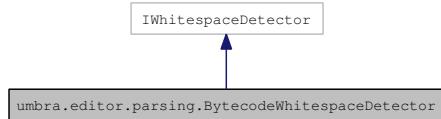
## 6.33 umbra.editor.parsing.BytecodeWhitespaceDetector Class Reference

This class defines objects that are able to check if a particular character is a whitespace.

Inheritance diagram for umbra.editor.parsing.BytecodeWhitespaceDetector:



Collaboration diagram for umbra.editor.parsing.BytecodeWhitespaceDetector:



### Public Member Functions

- final boolean `isWhitespace` (final char `a_char`)

*This method defines which characters are whitespace characters.*

#### 6.33.1 Detailed Description

This class defines objects that are able to check if a particular character is a whitespace.

##### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

##### Version:

a-01

#### 6.33.2 Member Function Documentation

##### 6.33.2.1 final boolean umbra.editor.parsing.BytecodeWhitespaceDetector.isWhitespace (final char `a_char`)

This method defines which characters are whitespace characters.

##### Parameters:

`a_char` the character to determine if it is whitespace

**Returns:**

  true when the character is regarded as a whitespace

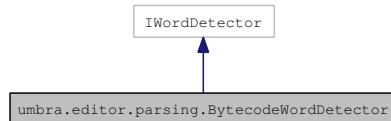
The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/[BytecodeWhitespaceDetector.java](#)

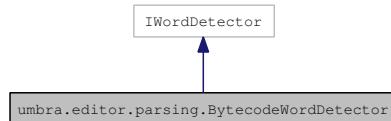
## 6.34 umbra.editor.parsing.BytecodeWordDetector Class Reference

The class implements the way the words are scanned in the Eclipse scanners used in the Umbra [editor](#).

Inheritance diagram for `umbra.editor.parsing.BytecodeWordDetector`:



Collaboration diagram for `umbra.editor.parsing.BytecodeWordDetector`:



### Public Member Functions

- final boolean `isWordStart` (final char `a_char`)  
*This method returns true when the character is a legal character to start a word.*
- final boolean `isWordPart` (final char `a_char`)  
*This method returns true when the character is a legal internal character of a word.*

#### 6.34.1 Detailed Description

The class implements the way the words are scanned in the Eclipse scanners used in the Umbra [editor](#).

##### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

##### Version:

a-01

#### 6.34.2 Member Function Documentation

##### 6.34.2.1 final boolean `umbra.editor.parsing.BytecodeWordDetector.isWordStart` (final char `a_char`)

This method returns true when the character is a legal character to start a word.

In this case it means it is a letter.

##### Parameters:

`a_char` a character to check

**Returns:**

true when the character can start a word, false otherwise

**6.34.2.2 final boolean umbra.editor.parsing.BytecodeWordDetector.isWordPart (final char *a\_char*)**

This method returns true when the character is a legal internal character of a word.

In this case it means it is a letter, a digit or an underscore sign ('\_').

**Parameters:**

*a\_char* a character to check

**Returns:**

true when the character can occur inside a word, false otherwise

**See also:**

[org.eclipse.jface.text.rules.IWordDetector.isWordPart\(char\)](#)

The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/[BytecodeWordDetector.java](#)

## 6.35 umbra.instructions.CannotCallRuleException Class Reference

This class is used to mark possible errors in quick dispatcher automaton [DispatchingAutomaton](#).

### Public Member Functions

- [CannotCallRuleException](#) (final Throwable *an\_ex*)

*This method creates the exception with the given reason that was handed in by some calculations before.*

### Static Private Attributes

- static final long [serialVersionUID](#) = 6117443445094038369L

*The serial ID for this class.*

### 6.35.1 Detailed Description

This class is used to mark possible errors in quick dispatcher automaton [DispatchingAutomaton](#).

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.35.2 Constructor & Destructor Documentation

#### 6.35.2.1 umbra.instructions.CannotCallRuleException.CannotCallRuleException (final Throwable *an\_ex*)

This method creates the exception with the given reason that was handed in by some calculations before.

#### Parameters:

*an\_ex* the exception which is the reason

### 6.35.3 Member Data Documentation

#### 6.35.3.1 final long umbra.instructions.CannotCallRuleException.serialVersionUID = 6117443445094038369L [static, private]

The serial ID for this class.

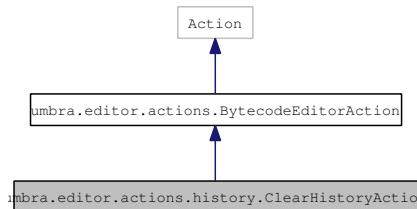
The documentation for this class was generated from the following file:

- source/umbra/instructions/[CannotCallRuleException.java](#)

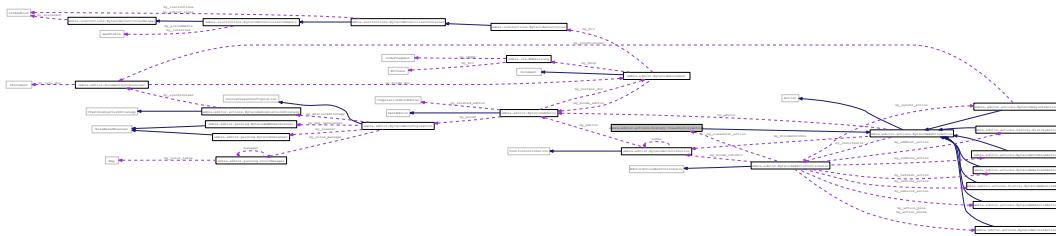
## 6.36 umbra.editor.actions.history.ClearHistoryAction Class Reference

The bytecode `editor` action that removes all the historical versions of code.

Inheritance diagram for `umbra.editor.actions.history.ClearHistoryAction`:



Collaboration diagram for `umbra.editor.actions.history.ClearHistoryAction`:



### Public Member Functions

- `ClearHistoryAction (final BytecodeEditorContributor a_contributor, final BytecodeContribution a_btcd_contribution)`  
*This constructor creates the action to add item to the `history` of the byte code `editor`.*
- `final void run ()`  
*This method clears the `history` for the currently active `editor`.*
- `void selectionChanged (final IAction an_action, final ISelection a_selection)`  
*The method reacts when the selected area changes in the bytecode `editor`.*

#### 6.36.1 Detailed Description

The bytecode `editor` action that removes all the historical versions of code.

##### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
 Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

##### Version:

a-01

## 6.36.2 Constructor & Destructor Documentation

### 6.36.2.1 `umbra.editor.actions.history.ClearHistoryAction` (final BytecodeEditorContributor *a\_contributor*, final BytecodeContribution *a\_btcd\_contribution*)

This constructor creates the action to add item to the [history](#) of the byte code [editor](#).

It registers the name of the action with the text "Clear history" and stores locally the object which creates all the [actions](#) and which contributes the [editor](#) GUI elements to the eclipse GUI.

#### Parameters:

*a\_contributor* the manager that initialises all the [actions](#) within the bytecode plugin

*a\_btcd\_contribution* the GUI elements contributed to the eclipse GUI by the bytecode [editor](#). This reference should be the same as in the parameter *a\_contributor*.

## 6.36.3 Member Function Documentation

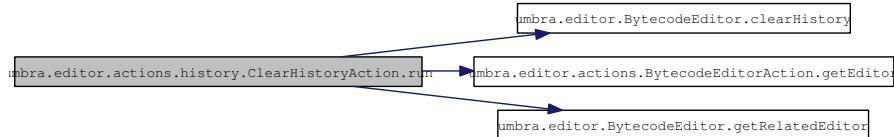
### 6.36.3.1 final void `umbra.editor.actions.history.ClearHistoryAction.run` ()

This method clears the [history](#) for the currently active [editor](#).

It resets the counter of the historical versions and then deletes all the files in the workspace that represent the historical versions of the current file.

References `umbra.editor.BytecodeEditor.clearHistory()`, `umbra.editor.actions.BytecodeEditorAction.getEditor()`, and `umbra.editor.BytecodeEditor.getRelatedEditor()`.

Here is the call graph for this function:



### 6.36.3.2 void `umbra.editor.actions.history.ClearHistoryAction.selectionChanged` (final IAction *an\_action*, final ISelection *a\_selection*)

The method reacts when the selected area changes in the bytecode [editor](#).

Currently, it does nothing.

#### Parameters:

*an\_action* the action which triggered the selection change

*a\_selection* the new selection.

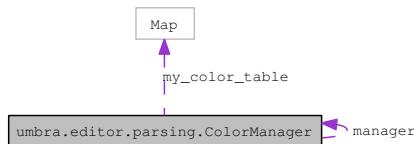
The documentation for this class was generated from the following file:

- source/umbra/editor/actions/history/[ClearHistoryAction.java](#)

## 6.37 umbra.editor.parsing.ColorManager Class Reference

This object manages the allocation and deallocation of the system colors that are used in the colouring in the bytecode editors.

Collaboration diagram for umbra.editor.parsing.ColorManager:



### Public Member Functions

- void [dispose \(\)](#)  
*This method disposes of the operating system resources associated with the colors in the bytecode editor.*
- Color [getColor \(final RGB a\\_rgb\)](#)  
*This method checks if the manager already has allocated the given value and in that case returns it.*

### Static Public Member Functions

- static [ColorManager getColorManager \(\)](#)  
*The static factory which returns the one and only [ColorManager](#) object in the running Umbra plugin.*

### Private Member Functions

- [ColorManager \(\)](#)  
*The private constructor to prevent creating objects otherwise than through the static factory method.*

### Private Attributes

- Map [my\\_color\\_table = new HashMap\(10\)](#)  
*This is a collection that remembers the values of all the already allocated colours.*

### Static Private Attributes

- static [ColorManager manager](#)  
*The one and only [ColorManager](#) object in the Umbra plugin.*

### 6.37.1 Detailed Description

This object manages the allocation and deallocation of the system colors that are used in the colouring in the bytecode editors.

**Author:**

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
 Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
 Wojciech Was ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

**Version:**

a-01

### 6.37.2 Constructor & Destructor Documentation

#### 6.37.2.1 `umbra.editor.parsing.ColorManager.ColorManager () [private]`

The private constructor to prevent creating objects otherwise than through the static factory method.

Referenced by `umbra.editor.parsing.ColorManager.getColorManager()`.

### 6.37.3 Member Function Documentation

#### 6.37.3.1 `static ColorManager umbra.editor.parsing.ColorManager.getColorManager () [static]`

The static factory which returns the one and only `ColorManager` object in the running Umbra plugin.

**Returns:**

the only color manager

References `umbra.editor.parsing.ColorManager.ColorManager()`, `umbra.editor.parsing.ColorManager.manager()` and `umbra.editor.parsing.ColorManager`.

Referenced by `umbra.editor.BytecodeConfiguration.BytecodeConfiguration()`.

Here is the call graph for this function:



#### 6.37.3.2 `void umbra.editor.parsing.ColorManager.dispose ()`

This method disposes of the operating system resources associated with the colors in the bytecode `editor`.

References `umbra.editor.parsing.ColorManager.my_color_table`.

Referenced by `umbra.editor.BytecodeConfiguration.disposeColor()`.

### 6.37.3.3 Color umbra.editor.parsing.ColorManager.getColor (final RGB *a\_rgb*)

This method checks if the menager already has allocated the given value and in that case returns it. In case the value has not been allocated yet, it allocates that from the system display.

#### Parameters:

*a\_rgb* the value of the colour to allocate

#### Returns:

the color object for the given RGB value

References umbra.editor.parsing.ColorManager.my\_color\_table.

Referenced by umbra.editor.parsing.TokenGetter.getTextAttribute().

## 6.37.4 Member Data Documentation

### 6.37.4.1 ColorManager umbra.editor.parsing.ColorManager.manager [static, private]

The one and only [ColorManager](#) object in the Umbra plugin.

Referenced by umbra.editor.parsing.ColorManager.getColorManager().

### 6.37.4.2 Map umbra.editor.parsing.ColorManager.my\_color\_table = new HashMap(10) [private]

This is a collection that remembers the values of all the already allocated colours.

This allows to reuse already allocated colours.

Referenced by umbra.editor.parsing.ColorManager.dispose(), and umbra.editor.parsing.ColorManager.getColor().

The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/[ColorManager.java](#)

## 6.38 umbra.editor.ColorModeContainer Class Reference

This class is a static container that keeps the value of current colouring style that is obtained after each refreshing (which takes place when a byte code document is created too).

### Static Public Member Functions

- static int [getMod \(\)](#)  
*This method returns the value of the current colouring style mode.*
- static void [setMod \(final int a\\_color\\_mode\)](#)  
*This method sets the value of the real colouring style.*
- static void [classKnown \(\)](#)  
*This method sets the mode of the current class to "class known".*
- static void [classUnknown \(\)](#)  
*This method sets the mode of the current class to "class unknown".*

### Private Member Functions

- [ColorModeContainer \(\)](#)  
*The empty constructor to forbid the creation of the instances.*

### Static Private Attributes

- static int [mod = 1](#)  
*The current value of the colouring style.*
- static boolean [disas](#)  
*The indicator of the normal and special modes.*

#### 6.38.1 Detailed Description

This class is a static container that keeps the value of current colouring style that is obtained after each refreshing (which takes place when a byte code document is created too).

This class has two modes of operation:

- The "class unknown" mode - in this case a special greyish colouring style is returned by the methods. This colouring indicates that the byte code has no connection with a class file so the editing will not change any class file.
- The "class known" mode - in this case a real colouring style is returned by the methods. This colouring indicates that the byte code has connection with a class file so the editing will change the corresponding class file. This mode is set on in moments when we know how to associate the class file to its textual representation.

Most of the time the class of a byte code textual representation that is fed to Umbra is not known so the default mode here is "class unknown" and the intent is to change this mode only for short periods when the class is indeed known.

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alex@smimuw.edu.pl](mailto:alex@smimuw.edu.pl))

**Version:**

a-01

## 6.38.2 Constructor & Destructor Documentation

### 6.38.2.1 umbra.editor.ColorModeContainer.ColorModeContainer () [private]

The empty constructor to forbid the creation of the instances.

## 6.38.3 Member Function Documentation

### 6.38.3.1 static int umbra.editor.ColorModeContainer.getMod () [static]

This method returns the value of the current colouring style mode.

In case the current class is in the normal state it returns the greyish style, in case the current class is in the special state it returns the real colouring style.

**Returns:**

the value of the colouring mode to be used

References umbra.editor.ColorModeContainer.disas, and umbra.editor.ColorModeContainer.mod.

### 6.38.3.2 static void umbra.editor.ColorModeContainer.setMod (final int *a\_color\_mode*) [static]

This method sets the value of the real colouring style.

**Parameters:**

*a\_color\_mode* the new value of the real colouring style

References umbra.editor.ColorModeContainer.mod.

### 6.38.3.3 static void umbra.editor.ColorModeContainer.classKnown () [static]

This method sets the mode of the current class to "class known".

References umbra.editor.ColorModeContainer.disas.

**6.38.3.4 static void umbra.editor.ColorModeContainer.classUnknown () [static]**

This method sets the mode of the current class to "class unknown".

References umbra.editor.ColorModeContainer.disas.

## 6.38.4 Member Data Documentation

**6.38.4.1 int umbra.editor.ColorModeContainer.mod = 1 [static, private]**

The current value of the colouring style.

The default colouring style number is 1.

Referenced by umbra.editor.ColorModeContainer.getMod(), and umbra.editor.ColorModeContainer.setMod().

**6.38.4.2 boolean umbra.editor.ColorModeContainer.disas [static, private]**

The indicator of the normal and special modes.

It is equal to `false` in case the mode is "class unknown" (greyish) and `true` in case the mode is "class known".

Referenced by umbra.editor.ColorModeContainer.classKnown(), umbra.editor.ColorModeContainer.classUnknown(), and umbra.editor.ColorModeContainer.getMod().

The documentation for this class was generated from the following file:

- source/umbra/editor/[ColorModeContainer.java](#)

## 6.39 umbra.editor.parsing.ColorValues Class Reference

The interface defining colours used in particular colouring styles.

### Static Public Attributes

- static final int **COMPONENT\_RED** = 0  
*The position of the red colour component in a single style entry from MODES\_DESC array.*
- static final int **COMPONENT\_GREEN** = 1  
*The position of the green colour component in a single style entry from MODES\_DESC array.*
- static final int **COMPONENT\_BLUE** = 2  
*The position of the blue colour component in a single style entry from MODES\_DESC array.*
- static final int **COMPONENT\_TXTSTYLE** = 3  
*The position of the font style component in a single style entry from MODES\_DESC array.*
- static final int **COMPONENT\_NUMBER** = COMPONENT\_TXTSTYLE + 1  
*The number of style parameters per slot.*
- static final int **SLOT\_STRING** = 0  
*The colour of strings.*
- static final int **SLOT\_COMMENT** = 1  
*The colour of comments.*
- static final int **SLOT\_DEFAULT** = 2  
*The colour of unparsed text in byte code (e.g.*
- static final int **SLOT\_ERROR** = 3  
*The colour of pieces of byte code recognized to be an error (not used).*
- static final int **SLOT\_HEADER** = 4  
*The colour of the method headers (e.g.*
- static final int **SLOT\_TAG** = 5  
*The colour of BML annotations.*
- static final int **SLOT\_BTAINSTR** = 7  
*The color of bytecode instructions.*
- static final int **SLOT\_KEY** = 8  
*The colour of the word: init.*
- static final int **SLOT\_LINE** = 9  
*The colour of the "LineNumber" areas.*
- static final int **SLOT\_THROWS** = 10

*The colour of the "Throws" areas.*

- static final int **SLOT\_PARENTHESES** = 11

*The colour of sections in byte code that are surrounded by '( )' or '{ }'.*

- static final int **SLOT\_NUMBER** = 12

*The color of numbers appearing in byte code except from cases listed below.*

- static final int **SLOT\_LABELNUMBER** = 13

*The colour of line number at the beginning of a line.*

- static final int **SLOT\_HASH** = 14

*The colour of number arguments that start with '#'.*

- static final int **SLOT\_PERCENT** = 15

*The colour of number arguments that start with "%".*

- static final int **SLOT\_BML** = 16

*The colour of the BML annotations.*

- static final int **SLOT\_BMLKEYWORDS** = 17

*The colour of keywords in the BML annotations.*

- static final int **SLOTS\_NO** = 18

*Number of defined colour constants.*

- static final int[ ][ ] **MODES\_DESC**

*The array which associates colour and text style modes with actual values of the RGB colours.*

## Private Member Functions

- **ColorValues ()**

*The private constructor to forbid the creation of objects with this type.*

### 6.39.1 Detailed Description

The interface defining colours used in particular colouring styles.

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
 Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

## 6.39.2 Constructor & Destructor Documentation

### 6.39.2.1 **umbra.editor.parsing.ColorValues()** [private]

The private constructor to forbid the creation of objects with this type.

## 6.39.3 Member Data Documentation

### 6.39.3.1 **final int umbra.editor.parsing.ColorValues.COMPONENT\_RED = 0** [static]

The position of the red colour component in a single style entry from [MODES\\_DESC](#) array.

### 6.39.3.2 **final int umbra.editor.parsing.ColorValues.COMPONENT\_GREEN = 1** [static]

The position of the green colour component in a single style entry from [MODES\\_DESC](#) array.

### 6.39.3.3 **final int umbra.editor.parsing.ColorValues.COMPONENT\_BLUE = 2** [static]

The position of the blue colour component in a single style entry from [MODES\\_DESC](#) array.

### 6.39.3.4 **final int umbra.editor.parsing.ColorValues.COMPONENT\_TXTSTYLE = 3** [static]

The position of the font style component in a single style entry from [MODES\\_DESC](#) array.

### 6.39.3.5 **final int umbra.editor.parsing.ColorValues.COMPONENT\_NUMBER = COMPONENT\_TXTSTYLE + 1** [static]

The number of style parameters per slot.

Currently, we have paramters for colour components red, green, blue and text style.

### 6.39.3.6 **final int umbra.editor.parsing.ColorValues.SLOT\_STRING = 0** [static]

The colour of strings.

### 6.39.3.7 **final int umbra.editor.parsing.ColorValues.SLOT\_COMMENT = 1** [static]

The colour of comments.

### 6.39.3.8 **final int umbra.editor.parsing.ColorValues.SLOT\_DEFAULT = 2** [static]

The colour of unparsed text in byte code (e.g.  
names of called methods).

### 6.39.3.9 **final int umbra.editor.parsing.ColorValues.SLOT\_ERROR = 3** [static]

The colour of pieces of byte code recognized to be an error (not used).

**6.39.3.10 final int umbra.editor.parsing.ColorValues.SLOT\_HEADER = 4 [static]**

The colour of the method headers (e.g.

public int a(int b)).

**6.39.3.11 final int umbra.editor.parsing.ColorValues.SLOT\_TAG = 5 [static]**

The colour of BML annotations.

**6.39.3.12 final int umbra.editor.parsing.ColorValues.SLOT\_BTAINSTR = 7 [static]**

The color of bytecode [instructions](#).

**6.39.3.13 final int umbra.editor.parsing.ColorValues.SLOT\_KEY = 8 [static]**

The colour of the word: init.

Currently unused.

**6.39.3.14 final int umbra.editor.parsing.ColorValues.SLOT\_LINE = 9 [static]**

The colour of the "LineNumber" areas.

FIXME: add handling of "Line" areas <https://mobius.ucd.ie/ticket/547>

**6.39.3.15 final int umbra.editor.parsing.ColorValues.SLOT\_THROWS = 10 [static]**

The colour of the "Throws" areas.

FIXME: add handling of "Line" areas <https://mobius.ucd.ie/ticket/549>

**6.39.3.16 final int umbra.editor.parsing.ColorValues.SLOT\_PARENTHESSES = 11 [static]**

The colour of sections in byte code that are surrounded by '()' or '{ }'.

**6.39.3.17 final int umbra.editor.parsing.ColorValues.SLOT\_NUMBER = 12 [static]**

The color of numbers appearing in byte code except from cases listed below.

**6.39.3.18 final int umbra.editor.parsing.ColorValues.SLOT\_LABELNUMBER = 13 [static]**

The colour of line number at the beginning of a line.

**6.39.3.19 final int umbra.editor.parsing.ColorValues.SLOT\_HASH = 14 [static]**

The colour of number arguments that start with '#'.

**6.39.3.20 final int umbra.editor.parsing.ColorValues.SLOT\_PERCENT = 15 [static]**

The colour of number arguments that start with ”.

**6.39.3.21 final int umbra.editor.parsing.ColorValues.SLOT\_BML = 16 [static]**

The colour of the BML annotations.

**6.39.3.22 final int umbra.editor.parsing.ColorValues.SLOT\_BMLKEYWORDS = 17 [static]**

The colour of keywords in the BML annotations.

**6.39.3.23 final int umbra.editor.parsing.ColorValues.SLOTS\_NO = 18 [static]**

Number of defined colour constants.

**6.39.3.24 final int [ ][] umbra.editor.parsing.ColorValues.MODES\_DESC [static]**

The array which associates colour and text style modes with actual values of the RGB colours.

The colouring mode is an index to the first coordinate of the array, the particular colour parameters are start at the position: slot\_number \* **COMPONENT\_NUMBER**. The style components are located at the following positions:

- slot\_number \* **COMPONENT\_NUMBER** + **COMPONENT\_RED**,
- slot\_number \* **COMPONENT\_NUMBER** + **COMPONENT\_GREEN**,
- slot\_number \* **COMPONENT\_NUMBER** + **COMPONENT\_BLUE**,
- slot\_number \* **COMPONENT\_NUMBER** + **COMPONENT\_TXTSTYLE**,

The available slots are: **SLOT\_STRING**, **SLOT\_COMMENT**, **SLOT\_DEFAULT**, **SLOT\_ERROR**, **SLOT\_HEADER**, **SLOT\_TAG**, **SLOT\_BTAINSTR**, **SLOT\_KEY**, **SLOT\_LINE**, **SLOT\_THROWS**, **SLOT\_PARENTHESSES**, **SLOT\_NUMBER**, **SLOT\_LABELNUMBER**, **SLOT\_HASH**, **SLOT\_PERCENT**, **SLOT\_BML**, **SLOT\_BMLKEYWORDS**.

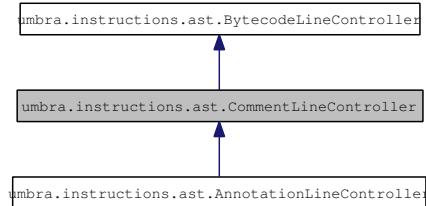
The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/[ColorValues.java](#)

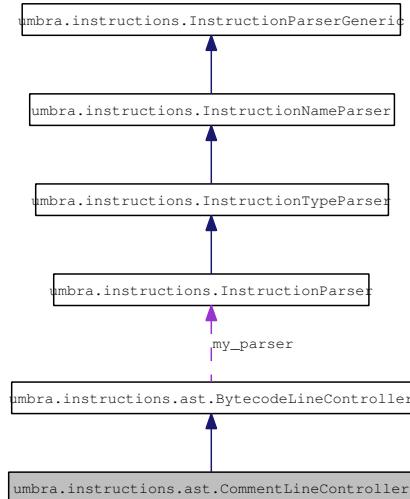
## 6.40 umbra.instructions.ast.CommentLineController Class Reference

This class handles the creation and correctness of line controllers that form comments.

Inheritance diagram for umbra.instructions.ast.CommentLineController:



Collaboration diagram for umbra.instructions.ast.CommentLineController:



### Public Member Functions

- [CommentLineController](#) (final String a\_line)  
*This constructor remembers only the line text with the comment content.*
- boolean [isCommentEnd](#) ()  
*Checks if the line can be an end of comment.*
- boolean [correct](#) ()  
*This method is used to check some basic condition of correctness.*

### Static Public Member Functions

- static boolean [isCommentStart](#) (finalString a\_line)

The method checks if the given string can be the start of a multi-line comment.

### 6.40.1 Detailed Description

This class handles the creation and correctness of line controllers that form comments.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.40.2 Constructor & Destructor Documentation

#### 6.40.2.1 umbra.instructions.ast.CommentLineController.CommentLineController (final String *a\_line*)

This constructor remembers only the line text with the comment content.

**Parameters:**

*a\_line* the string representation of the line for the line with comments

**See also:**

[BytecodeLineController](#).[BytecodeLineController](#)(String)

### 6.40.3 Member Function Documentation

#### 6.40.3.1 static boolean umbra.instructions.ast.CommentLineController.isCommentStart (final String *a\_line*) [static]

The method checks if the given string can be the start of a multi-line comment.

We use the heuristic that the line must start with "/\*" possibly with some initial whitespace before the sequence.

**Parameters:**

*a\_line* the string to be checked

**Returns:**

`true` when the string can start comment

#### 6.40.3.2 boolean umbra.instructions.ast.CommentLineController.isCommentEnd ()

Checks if the line can be an end of comment.

This holds when the final non-whitespace sequence in the line is \* / string.

**Returns:**

true when the line contains the end of comment sequence, false otherwise

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References [umbra.instructions.ast.BytecodeLineController.getMy\\_line\\_text\(\)](#).

Referenced by [umbra.instructions.Preparsing.getType\(\)](#).

Here is the call graph for this function:



#### 6.40.3.3 boolean [umbra.instructions.ast.CommentLineController.correct\(\)](#)

This method is used to check some basic condition of correctness.

For comment lines this is always true.

**Returns:**

true if the instruction is correct

**See also:**

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

Reimplemented in [umbra.instructions.ast.AnnotationLineController](#).

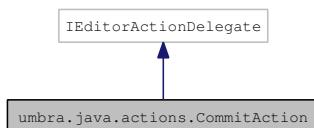
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[CommentLineController.java](#)

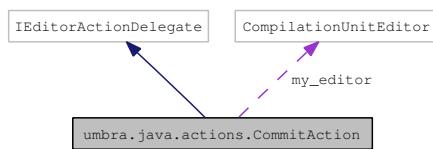
## 6.41 umbra.java.actions.CommitAction Class Reference

The action is used to commit changes made to Java source code.

Inheritance diagram for umbra.java.actions.CommitAction:



Collaboration diagram for umbra.java.actions.CommitAction:



### Public Member Functions

- final void  [setActiveEditor](#) (final IAction an\_action, final IEeditorPart a\_target\_editor)  
*The method saves the [editor](#) for the Java code file.*
- final void  [run](#) (final IAction an\_action)  
*This method is invoked when the Umbra "Commit" button is pressed in a Java file [editor](#).*
- void  [selectionChanged](#) (final IAction an\_action, final ISelection a\_selection)  
*The method reacts when the selected area changes in the Java source code [editor](#).*

### Private Attributes

- CompilationUnitEditor [my\\_editor](#)  
*The [editor](#) for the corresponding Java file.*

#### 6.41.1 Detailed Description

The action is used to commit changes made to Java source code.

After running it the rebuild action will create a byte code related to the committed version.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

## 6.41.2 Member Function Documentation

### 6.41.2.1 final void umbra.java.actions.CommitAction.setActiveEditor (final IAction *an\_action*, final IEeditorPart *a\_target\_editor*)

The method saves the [editor](#) for the Java code file.

#### Parameters:

*an\_action* the GUI action which triggered the [editor](#) change

*a\_target\_editor* the [editor](#) of the Java source code file

References [umbra.java.actions.CommitAction.my\\_editor](#).

### 6.41.2.2 final void umbra.java.actions.CommitAction.run (final IAction *an\_action*)

This method is invoked when the Umbra "Commit" button is pressed in a Java file [editor](#).

It saves the current Java file and deletes from workspace the original class file which contains the result of Java compilation (

#### See also:

[BytecodeEditor.doSave\(IProgressMonitor\)](#).

#### Parameters:

*an\_action* the action that triggered the operation

#### See also:

[org.eclipse.ui.IActionDelegate.run\(IAction\)](#)

References [umbra.java.actions.CommitAction.my\\_editor](#).

### 6.41.2.3 void umbra.java.actions.CommitAction.selectionChanged (final IAction *an\_action*, final ISelection *a\_selection*)

The method reacts when the selected area changes in the Java source code [editor](#).

Currently, it does nothing.

#### Parameters:

*an\_action* the action which triggered the selection change

*a\_selection* the new selection

## 6.41.3 Member Data Documentation

### 6.41.3.1 CompilationUnitEditor **umbra.java.actions.CommitAction.my\_editor** [private]

The [editor](#) for the corresponding Java file.

Referenced by [umbra.java.actions.CommitAction.run\(\)](#), and [umbra.java.actions.CommitAction.setActiveEditor\(\)](#).

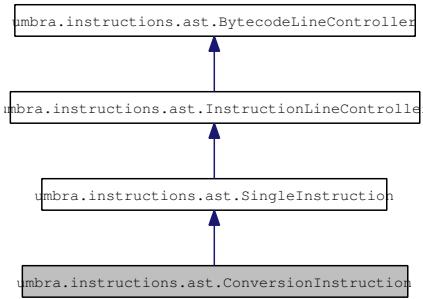
The documentation for this class was generated from the following file:

- source/umbra/java/actions/[CommitAction.java](#)

## 6.42 umbra.instructions.ast.ConversionInstruction Class Reference

This class handles the creation and correctness for the [instructions](#) with no parameters which convert types.

Inheritance diagram for `umbra.instructions.ast.ConversionInstruction`:



Collaboration diagram for `umbra.instructions.ast.ConversionInstruction`:



### Public Member Functions

- `ConversionInstruction (final String a_line_text, final String a_name)`  
*This creates an instance of an instruction named as `a_name` with the line text `a_line`.*
- `boolean correct ()`  
*Conversion instruction line is correct if it has no parameter.*
- `Instruction getInstruction ()`  
*This method, based on the value of the mnemonic name, creates a new BCEL instruction object for an instruction with no parameters.*

### Static Public Member Functions

- `static String[] getMnemonics ()`  
*This method returns the array of conversion [instructions](#) mnemonics.*

### Private Member Functions

- `Instruction getL2XConvOp (final Instruction a_res)`  
*This method creates the objects that represent [instructions](#) that convert values from the long type.*
- `Instruction getI2XConvOp (final Instruction a_res)`  
*This method creates the objects that represent [instructions](#) that convert values from the int type.*

- Instruction [getF2XConvOp](#) (final Instruction a\_res)

*This method creates the objects that represent [instructions](#) that convert values from the float type.*

- Instruction [getD2XConvOp](#) (final Instruction a\_res)

*This method creates the objects that represent [instructions](#) that convert values from the double type.*

### 6.42.1 Detailed Description

This class handles the creation and correctness for the [instructions](#) with no parameters which convert types.

The [instructions](#) handled here are:

- conversion from doubles,
- conversion from floats,
- conversion from integers,
- conversion from longs.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.42.2 Constructor & Destructor Documentation

#### 6.42.2.1 umbra.instructions.ast.ConversionInstruction.ConversionInstruction (final String a\_line\_text, final String a\_name)

This creates an instance of an instruction named as a\_name with the line text a\_line.

Currently it just calls the constructor of the superclass.

**Parameters:**

- a\_line\_text** the line number of the instruction
- a\_name** the mnemonic name of the instruction

**See also:**

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.42.3 Member Function Documentation

#### 6.42.3.1 static String [ ] umbra.instructions.ast.ConversionInstruction.getMnemonics () [static]

This method returns the array of conversion [instructions](#) mnemonics.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController.getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

#### 6.42.3.2 boolean [umbra.instructions.ast.ConversionInstruction.correct\(\)](#)

Conversion instruction line is correct if it has no parameter.

That means this must have the form: whitespace number : whitespace mnemonic whitespace lineend where mnemonic comes from [BytecodeStrings#SINGLE\\_INS](#).

**Returns:**

`true` when the instruction mnemonic is the only text in the line of the instruction text

**See also:**

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

Referenced by [umbra.instructions.ast.ConversionInstruction.getInstruction\(\)](#).

#### 6.42.3.3 Instruction [umbra.instructions.ast.ConversionInstruction.getInstruction\(\)](#)

This method, based on the value of the mnemonic name, creates a new BCEL instruction object for an instruction with no parameters.

The method can construct the following kinds of [instructions](#):

- conversion from doubles,
- conversion from floats,
- conversion from integers,
- conversion from longs.

This method also checks the syntactical correctness of the current instruction line.

**Returns:**

the freshly constructed BCEL instruction or `null` in case the instruction is not a stack instruction and in case the instruction line is incorrect

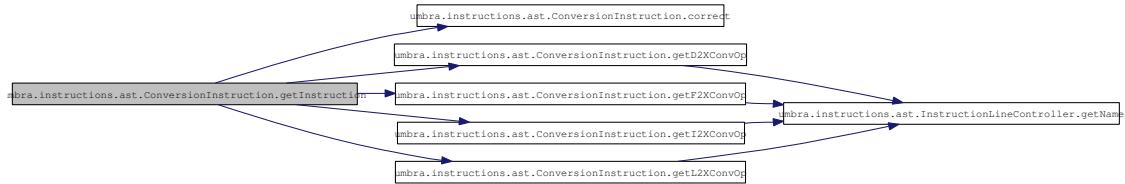
**See also:**

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

References [umbra.instructions.ast.ConversionInstruction.correct\(\)](#), [umbra.instructions.ast.ConversionInstruction.getD2XConvOp\(\)](#), [umbra.instructions.ast.ConversionInstruction.getF2XConvOp\(\)](#), [umbra.instructions.ast.ConversionInstruction.getI2XConvOp\(\)](#), [umbra.instructions.ast.ConversionInstruction.getL2XConvOp\(\)](#), and [umbra.instructions.ast.ConversionInstruction.getName\(\)](#).

Here is the call graph for this function:



#### 6.42.3.4 Instruction [umbra.instructions.ast.ConversionInstruction.getL2XConvOp](#) (final Instruction *a\_res*) [private]

This method creates the objects that represent [instructions](#) that convert values from the long type.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The [instructions](#) to convert from the long type are:

- l2d,
- l2f,
- l2i.

##### Parameters:

*a\_res* a helper value returned in case the current instruction is not in the current set

##### Returns:

the object that represents the current instruction or *res* in case the current instruction is not in the current set

References [umbra.instructions.ast.InstructionLineController.getName\(\)](#).

Referenced by [umbra.instructions.ast.ConversionInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



#### 6.42.3.5 Instruction [umbra.instructions.ast.ConversionInstruction.getI2XConvOp](#) (final Instruction *a\_res*) [private]

This method creates the objects that represent [instructions](#) that convert values from the int type.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The [instructions](#) to convert from the int type are:

- `i2b`,
- `i2c`,
- `i2d`,
- `i2f`,
- `i2l`,
- `i2s`.

**Parameters:**

`a_res` a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.ConversionInstruction.getInstruction()`.

Here is the call graph for this function:



#### 6.42.3.6 Instruction `umbra.instructions.ast.ConversionInstruction.getF2XConvOp` (final Instruction `a_res`) [private]

This method creates the objects that represent [instructions](#) that convert values from the float type.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The [instructions](#) to convert from the float type are:

- `f2d`,
- `f2i`,
- `f2l`.

**Parameters:**

`a_res` a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.ConversionInstruction.getInstruction()`.

Here is the call graph for this function:



#### 6.42.3.7 Instruction `umbra.instructions.ast.ConversionInstruction.getD2XConvOp (final Instruction a_res) [private]`

This method creates the objects that represent `instructions` that convert values from the double type.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The `instructions` to convert from the double type are:

- `d2f`,
- `d2i`,
- `d2l`.

##### Parameters:

`a_res` a helper value returned in case the current instruction is not in the current set

##### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.ConversionInstruction.getInstruction()`.

Here is the call graph for this function:



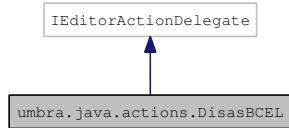
The documentation for this class was generated from the following file:

- `source/umbra/instructions/ast/ConversionInstruction.java`

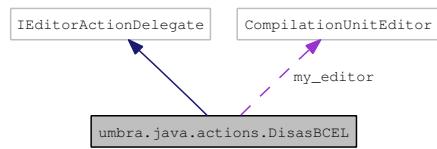
## 6.43 umbra.java.actions.DisasBCEL Class Reference

This class defines the action related to Java source `editor`.

Inheritance diagram for `umbra.java.actions.DisasBCEL`:



Collaboration diagram for `umbra.java.actions.DisasBCEL`:



### Public Member Functions

- final void `run` (final IAction an\_action)  
*Finds `org.apache.bcel.classfile.JavaClass` structure related to the current Java source code.*
- void `selectionChanged` (final IAction an\_action, final ISelection a\_selection)  
*Currently, does nothing.*
- final void `setActiveEditor` (final IAction an\_action, final IEIPart a\_target\_editor)  
*It sets the `editor` with the Java source code.*

### Private Member Functions

- IPath `openBCodeEditorForJavaFile` (final IFile a\_jfile) throws PartInitException, JavaModelException, ClassNotFoundException  
*This method opens a byte code `editor` for the given file that corresponds to a Java resource.*
- void `messageClassNotFound` (final IPath a\_path)  
*This method opens a warning dialog with the information that the given path does not exist.*
- boolean `checkJavaExtension` ()  
*This method checks if the source code `editor` edits `.java` file.*
- boolean `checkIfSaveNeeded` ()  
*This method checks if the source code `editor` must be saved before an action can be performed.*
- void `openEditorAndDisassemble` (final IWorkbenchPage a\_page, final BytecodeEditor an\_editor, final FileEditorInput an\_input, final BytecodeDocument a\_doc)  
*This method changes the colouring mode of a previously opened `editor`.*

## Private Attributes

- CompilationUnitEditor `my_editor`

The `editor` of a Java file for which the byte code file is generated.

### 6.43.1 Detailed Description

This class defines the action related to Java source `editor`.

Its execution causes generating new related byte code file in a new `editor` window.

#### Author:

BYTECODE team (contact `alex@mimuw.edu.pl`)

#### Version:

a-01

### 6.43.2 Member Function Documentation

#### 6.43.2.1 final void umbra.java.actions.DisasBCEL.run (final IAction *an\_action*)

Finds `org.apache.bcel.classfile.JavaClass` structure related to the current Java source code.

Generates new byte code from it and displays it in a new byte code `editor` window.

#### Parameters:

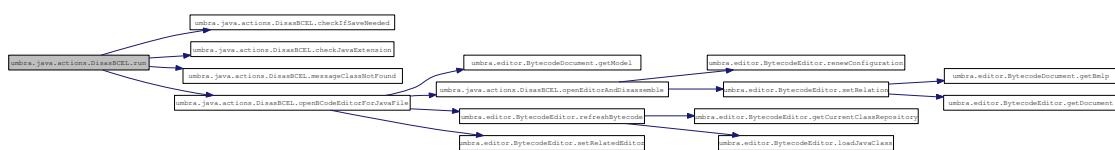
*an\_action* see the `IActionDelegate.run(IAction)`

#### See also:

`org.eclipse.ui.IActionDelegate.run(IAction)`

References `umbra.java.actions.DisasBCEL.checkIfSaveNeeded()`, `umbra.java.actions.DisasBCEL.checkJavaExtension()`, `umbra.java.actions.DisasBCEL.messageClassNotFound()`, `umbra.java.actions.DisasBCEL.my_editor`, and `umbra.java.actions.DisasBCEL.openBCodeEditorForJavaFile()`.

Here is the call graph for this function:



#### 6.43.2.2 IPath umbra.java.actions.DisasBCEL.openBCodeEditorForJavaFile (final IFile *a\_jfile*) throws PartInitException, JavaModelException, ClassNotFoundException [private]

This method opens a byte code `editor` for the given file that corresponds to a Java resource.

It figures out the name of the .btc file and opens a byte code [editor](#) for this file. Subsequently it retrieves the corresponding document and does the refresh of the byte code contained in the document. This operation generates the textual content of the document. Next the current method regenerates the colouring of the document so that the document is not gray. At last the fresh content of the document is saved to the .btc file on disc

**Parameters:**

*a\_jfile* the file with a path to the Java resource

**Returns:**

the path of the class file from which the textual representation was generated

**Exceptions:**

*PartInitException* in case the byte code [editor](#) cannot be open

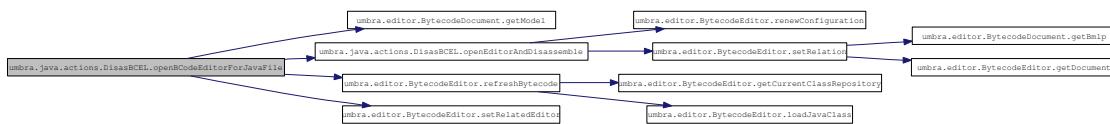
*JavaModelException* in case the current project has no class file output location set

*ClassNotFoundException* in case the class file for the given Java file cannot be found

References      `umbra.editor.BytecodeDocument.getModel()`,      `umbra.java.actions.DisasBCEL.my_editor`,      `umbra.java.actions.DisasBCEL.openEditorAndDisassemble()`,      `umbra.editor.BytecodeEditor.refreshBytecode()`, and `umbra.editor.BytecodeEditor.setRelatedEditor()`.

Referenced by `umbra.java.actions.DisasBCEL.run()`.

Here is the call graph for this function:



#### 6.43.2.3 void `umbra.java.actions.DisasBCEL.messageClassNotFound (final IPPath a_path)` [private]

This method opens a warning dialog with the information that the given path does not exist.

**Parameters:**

*a\_path* the path which does not exist

References `umbra.java.actions.DisasBCEL.my_editor`.

Referenced by `umbra.java.actions.DisasBCEL.run()`.

#### 6.43.2.4 boolean `umbra.java.actions.DisasBCEL.checkJavaExtension ()` [private]

This method checks if the source code [editor](#) edits `.java` file.

In case the file is not a `.java` file a popup with appropriate message is shown.

**Returns:**

`true` if the file is not a `.java` file, `false` otherwise

References umbra.java.actions.DisasBCEL.my\_editor.

Referenced by umbra.java.actions.DisasBCEL.run().

#### 6.43.2.5 boolean umbra.java.actions.DisasBCEL.checkIfSaveNeeded () [private]

This method checks if the source code [editor](#) must be saved before an action can be performed.

In case the [editor](#) must be saved a popup with appropriate message is shown.

**Returns:**

`true` when the save is needed, `false` otherwise

References umbra.java.actions.DisasBCEL.my\_editor.

Referenced by umbra.java.actions.DisasBCEL.run().

#### 6.43.2.6 void umbra.java.actions.DisasBCEL.openEditorAndDisassemble (final IWorkbenchPage *a\_page*, final BytecodeEditor *an\_editor*, final FileEditorInput *an\_input*, final BytecodeDocument *a\_doc*) [private]

This method changes the colouring mode of a previously opened [editor](#).

Now, the content of the [editor](#) is coloured with the current colouring style instead of the gray style which is the default for documents with no connection with a class file.

**Parameters:**

*a\_page* a workbench page in which the new [editor](#) is reconfigured

*an\_editor* the [editor](#) to change the colouring for

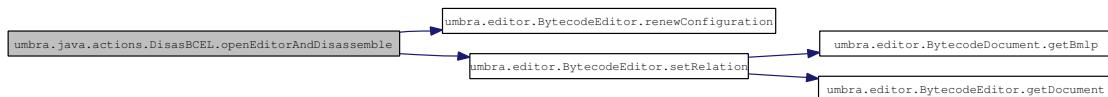
*an\_input* an input which will be presented in the [editor](#)

*a\_doc* a document where the BCEL and BMLlib connection is already set

References umbra.java.actions.DisasBCEL.my\_editor, umbra.editor.BytecodeEditor.renewConfiguration(), and umbra.editor.BytecodeEditor.setRelation().

Referenced by umbra.java.actions.DisasBCEL.openBCodeEditorForJavaFile().

Here is the call graph for this function:



#### 6.43.2.7 void umbra.java.actions.DisasBCEL.selectionChanged (final IAction *an\_action*, final ISelection *a\_selection*)

Currently, does nothing.

**Parameters:**

*an\_action* see [org.eclipse.ui.IActionDelegate#selectionChanged\(IAction,ISelection\)](#)

*a\_selection* see [org.eclipse.ui.IActionDelegate#selectionChanged\(IAction,ISelection\)](#)

#### 6.43.2.8 final void umbra.java.actions.DisasBCEL.setActiveEditor (final IAction *an\_action*, final IEeditorPart *a\_target\_editor*)

It sets the [editor](#) with the Java source code.

**Parameters:**

*an\_action* see [IEditorPart](#))

*a\_target\_editor* the new [editor](#) to be active for the action

References umbra.java.actions.DisasBCEL.my\_editor.

### 6.43.3 Member Data Documentation

#### 6.43.3.1 CompilationUnitEditor umbra.java.actions.DisasBCEL.my\_editor [private]

The [editor](#) of a Java file for which the byte code file is generated.

Referenced by umbra.java.actions.DisasBCEL.checkIfSaveNeeded(), umbra.java.actions.DisasBCEL.checkJavaExtension(), umbra.java.actions.DisasBCEL.messageClassNotFound(), umbra.java.actions.DisasBCEL.openBCodeEditorForJavaFile(), umbra.java.actions.DisasBCEL.openEditorAndDisassemble(), umbra.java.actions.DisasBCEL.run(), and umbra.java.actions.DisasBCEL.setActiveEditor().

The documentation for this class was generated from the following file:

- source/umbra/java/actions/[DisasBCEL.java](#)

## 6.44 umbra.instructions.DispatchingAutomaton Class Reference

This class implements an automaton which is used to quickly determine the type of the currently analysed line of the byte code text.

### Public Member Functions

- [DispatchingAutomaton \(\)](#)

*This constructor creates the automaton such that the default rule is executed and the set of outgoing edges is empty.*

### Private Attributes

- TreeMap< Character, [DispatchingAutomaton](#) > [my\\_outgoing](#)

*This field represents the set of the outgoing edges from the current node of the automaton.*

- Class [my\\_rule](#)

*This is the class which should be created in case the given parsed string points to the current node.*

- String [my\\_mnemonic](#)

*The string which holds the mnemonic to be used for the instruction lines.*

### Static Private Attributes

- static final Class [DEFAULT\\_RULE](#) = UnknownLineController.class

*In case no particular rule was set for this node, the automaton creates an object of this class.*

### 6.44.1 Detailed Description

This class implements an automaton which is used to quickly determine the type of the currently analysed line of the byte code text.

The automaton is constructed out of nodes each of which has the outgoing edges labelled with characters ([Character](#)). One can do the following operations on the automaton:

- add a rule to create a [BytecodeLineController](#) when a terminal node of the automaton is reached,
- add a loop rule which allows the automaton to move along star-like regular expressions,
- execute the rule for the given string line.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

## 6.44.2 Constructor & Destructor Documentation

### 6.44.2.1 **umbra.instructions.DispatchingAutomaton.DispatchingAutomaton ()**

This constructor creates the automaton such that the default rule is executed and the set of outgoing edges is empty.

References

umbra.instructions.DispatchingAutomaton.DEFAULT\_RULE,  
umbra.instructions.DispatchingAutomaton.my\_outgoing, and  
umbra.instructions.DispatchingAutomaton.my\_rule.

## 6.44.3 Member Data Documentation

### 6.44.3.1 **final Class umbra.instructions.DispatchingAutomaton.DEFAULT\_RULE = UnknownLineController.class [static, private]**

In case no particular rule was set for this node, the automaton creates an object of this class.

Referenced by umbra.instructions.DispatchingAutomaton.DispatchingAutomaton().

### 6.44.3.2 **TreeMap< Character, DispatchingAutomaton > umbra.instructions.DispatchingAutomaton.my\_outgoing [private]**

This field represents the set of the outgoing edges from the current node of the automaton.

Referenced by umbra.instructions.DispatchingAutomaton.DispatchingAutomaton().

### 6.44.3.3 **Class umbra.instructions.DispatchingAutomaton.my\_rule [private]**

This is the class which should be created in case the given parsed string points to the current node.

Referenced by umbra.instructions.DispatchingAutomaton.DispatchingAutomaton().

### 6.44.3.4 **String umbra.instructions.DispatchingAutomaton.my\_mnemonic [private]**

The string which holds the mnemonic to be used for the instruction lines.

It is used only when it is set to be non-null.

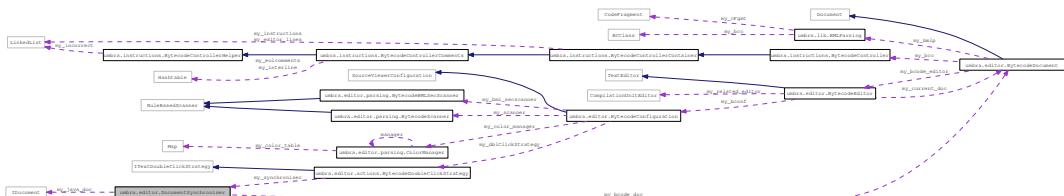
The documentation for this class was generated from the following file:

- source/umbra/instructions/[DispatchingAutomaton.java](#)

## 6.45 umbra.editor.DocumentSynchroniser Class Reference

This class handles the logic of the synchronisation of the cursor positions between the source code and the byte code documents.

## Collaboration diagram for umbra.editor.DocumentSynchroniser:



## Public Member Functions

- `DocumentSynchroniser` (final BytecodeDocument a\_bdoc, final IDocument a\_jdoc)

*The constructor initialises the relation between the byte code document and the source code document to make the synchronisation with.*

- final void **synchronizeBS** (final int a\_pos) throws UmbraLocationException, UmbraSynchronisationException

*Highlights the area in the source code editor which corresponds to the marked area in the byte code editor.*

## Private Member Functions

- int syncBS (final JavaClass a\_java\_class, final int a\_line\_no) throws UmbraException, UmbraSynchonisationException

*Computes the area in current Java source code corresponding to given line of the byte code document.*

## Private Attributes

- `BytecodeDocument my_bcode_doc`

*The byte code document which takes part in the synchronisation process.*

- IDocument my\_java\_doc

*The Java source code document which takes part in the synchronisation process.*

## Static Private Attributes

- static final int NO\_OF\_POSITIONS = 2

This is the size of the array which contains the range of positions of the target document (e.g.

### 6.45.1 Detailed Description

This class handles the logic of the synchronisation of the cursor positions between the source code and the byte code documents.

It computes for a given source code line a corresponding byte code line and for a given byte code line the corresponding source code line range. It uses the class file line number table to perform these operations.

**Author:**

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.45.2 Constructor & Destructor Documentation

#### 6.45.2.1 **umbra.editor.DocumentSynchroniser.DocumentSynchroniser (final BytecodeDocument *a\_bdoc*, final IDocument *a\_jdoc*)**

The constructor initialises the relation between the byte code document and the source code document to make the synchronisation with.

**Parameters:**

*a\_bdoc* the byte code document  
*a\_jdoc* the Java source code document

References `umbra.editor.DocumentSynchroniser.my_bcode_doc`, and `umbra.editor.DocumentSynchroniser.my_java_doc`.

### 6.45.3 Member Function Documentation

#### 6.45.3.1 **final void umbra.editor.DocumentSynchroniser.synchronizeBS (final int *a\_pos*) throws UmbraLocationException, UmbraSynchronisationException**

Highlights the area in the source code `editor` which corresponds to the marked area in the byte code `editor`.

Works correctly only inside a method body.

**See also:**

`DocumentSynchroniser.synchronizeSB(int, CompilationUnitEditor)`

**Parameters:**

*a\_pos* index of line in byte code `editor`. Lines in related source code `editor` corresponding to this line will be highlighted.

**Exceptions:**

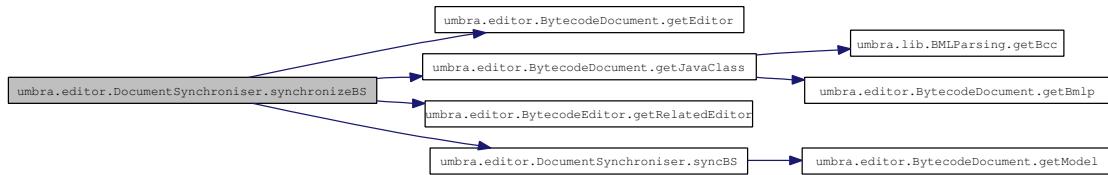
`UmbraLocationException` in case a position is reached in the source code or byte code `editor` which does not exists there

***UmbraSynchronisationException*** in case there is no instruction line which can be reasonably associated with the given position

References `umbra.editor.BytecodeDocument.getEditor()`, `umbra.editor.BytecodeDocument.getJavaClass()`, `umbra.editor.BytecodeEditor.getRelatedEditor()`, `umbra.editor.DocumentSynchroniser.my_bcode_doc`, `umbra.editor.DocumentSynchroniser.my_java_doc`, and `umbra.editor.DocumentSynchroniser.syncBS()`.

Referenced by `umbra.editor.actions.BytecodeDoubleClickStrategy.doubleClicked()`, `umbra.editor.actions.BytecodeSynchrAction.run()`, and `umbra.editor.actions.BytecodeSynchrAction.synchronizeBS()`.

Here is the call graph for this function:



#### 6.45.3.2 int umbra.editor.DocumentSynchroniser.syncBS (final JavaClass *a\_java\_class*, final int *a\_line\_no*) throws UmbraException, UmbraSynchronisationException [private]

Computes the area in current Java source code corresponding to given line of the byte code document.

The byte code should be refreshed before calling this method to update JavaClass structures. Works correctly only inside a method.

Algorithm:

- We obtain the number of the first instruction line not above the given position (we synchronise the BML annotations and comments so that the instruction below them is considered to be a pointer to the source code).
- We obtain the number of the method which contains the line (to be able to use the LineNumberTable).
- We retrieve the label of the instruction line we found (as the positions in the LineNumberTable are indexed with the labels).
- We look for the highest byte code label number in the LineNumberTable which is lower than the one we have already found (the entries in show where the source code line number changes, if there is no current label there then the current source code line begins at line with some lower label).
- We return the source code line from this entry in the LineNumberTable

#### Parameters:

*a\_java\_class* [org.apache.bcel.classfile.JavaClass](#) with current byte code BCEL representation  
*a\_line\_no* index of line in byte code [editor](#)

#### Returns:

the line of the source code corresponding to given byte code line)

#### Exceptions:

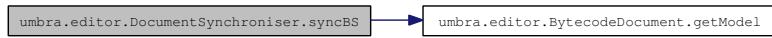
***UmbraException*** in case there is no instruction line that can be reasonably associated with the given line number

***UmbraSynchronisationException*** in case there is no instruction line that can be reasonably associated with the given line number

References `umbra.editor.BytecodeDocument.getModel()`, and `umbra.editor.DocumentSynchroniser.my_bcode_doc`.

Referenced by `umbra.editor.DocumentSynchroniser.syncBS()`.

Here is the call graph for this function:



#### 6.45.4 Member Data Documentation

##### 6.45.4.1 final int umbra.editor.DocumentSynchroniser.NO\_OF\_POSITIONS = 2 [static, private]

This is the size of the array which contains the range of positions of the target document (e.g. source code) that corresponds to the initial position in the initial document (e.g. byte code).

##### 6.45.4.2 BytecodeDocument umbra.editor.DocumentSynchroniser.my\_bcode\_doc [private]

The byte code document which takes part in the synchronisation process.

Referenced by `umbra.editor.DocumentSynchroniser.DocumentSynchroniser()`, `umbra.editor.DocumentSynchroniser.syncBS()`, and `umbra.editor.DocumentSynchroniser.synchronizeBS()`.

##### 6.45.4.3 IDocument umbra.editor.DocumentSynchroniser.my\_java\_doc [private]

The Java source code document which takes part in the synchronisation process.

Referenced by `umbra.editor.DocumentSynchroniser.DocumentSynchroniser()`, and `umbra.editor.DocumentSynchroniser.synchronizeBS()`.

The documentation for this class was generated from the following file:

- `source/umbra/editor/DocumentSynchroniser.java`

## 6.46 umbra.lib.EclipseIdentifiers Class Reference

This is just a container for textual Eclipse identifiers of objects defined in Umbra.

### Static Public Attributes

- static final String **BYTECODE\_EDITOR\_CLASS**  
*The text `editor` extension identifier which identifies the Umbra bytecode `editor`.*
- static final String **EOL** = System.getProperty("line.separator")  
*The end of line sequence for the current operating system.*

### Private Member Functions

- **EclipseIdentifiers ()**  
*The empty constructor to forbid the creation of the instances.*

#### 6.46.1 Detailed Description

This is just a container for textual Eclipse identifiers of objects defined in Umbra.

FIXME: it does not contain all the identifiers around <https://mobius.ucd.ie/ticket/590>

##### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

##### Version:

a-01

#### 6.46.2 Constructor & Destructor Documentation

##### 6.46.2.1 umbra.lib.EclipseIdentifiers.EclipseIdentifiers () [private]

The empty constructor to forbid the creation of the instances.

#### 6.46.3 Member Data Documentation

##### 6.46.3.1 final String umbra.lib.EclipseIdentifiers.BYTECODE\_EDITOR\_CLASS [static]

##### Initial value:

`"umbra.BytecodeEditor"`

The text `editor` extension identifier which identifies the Umbra bytecode `editor`.

**6.46.3.2 final String umbra.lib.EclipseIdentifiers.EOL = System.getProperty("line.separator")**  
[static]

The end of line sequence for the current operating system.

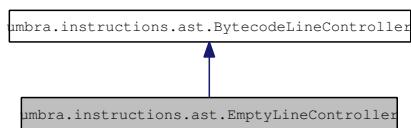
The documentation for this class was generated from the following file:

- source/umbra/lib/[EclipseIdentifiers.java](#)

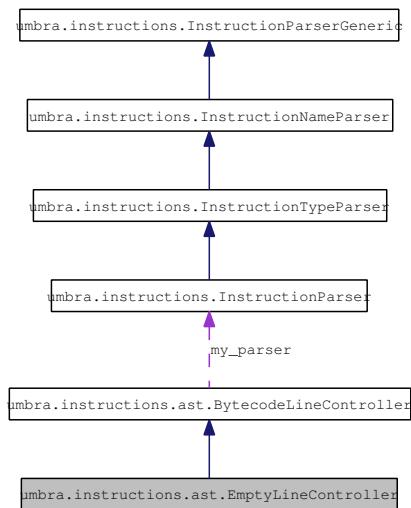
## 6.47 umbra.instructions.ast.EmptyLineController Class Reference

This is a class for a line with whitespaces only.

Inheritance diagram for umbra.instructions.ast.EmptyLineController:



Collaboration diagram for umbra.instructions.ast.EmptyLineController:



### Public Member Functions

- [EmptyLineController \(final String a\\_line\\_text\)](#)

*This constructor remembers only the line text of the line which contains solely whitespaces.*

- final boolean [correct \(\)](#)

#### 6.47.1 Detailed Description

This is a class for a line with whitespaces only.

##### Author:

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

##### Version:

a-01

## 6.47.2 Constructor & Destructor Documentation

### 6.47.2.1 **umbra.instructions.ast.EmptyLineController.EmptyLineController (final String *a\_line\_text*)**

This constructor remembers only the line text of the line which contains solely whitespaces.

**Parameters:**

*a\_line\_text* the string representation of the line

**See also:**

[BytecodeLineController.BytecodeLineController\(String\)](#)

## 6.47.3 Member Function Documentation

### 6.47.3.1 **final boolean umbra.instructions.ast.EmptyLineController.correct ()**

**Returns:**

true - an empty line is always correct

**See also:**

[BytecodeLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

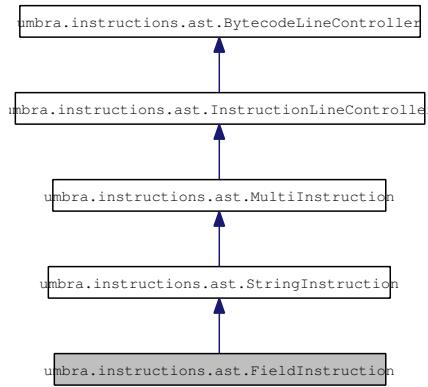
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[EmptyLineController.java](#)

## 6.48 umbra.instructions.ast.FieldInstruction Class Reference

This class handles the creation and correctness for [instructions](#) to store and load field values.

Inheritance diagram for `umbra.instructions.ast.FieldInstruction`:



Collaboration diagram for `umbra.instructions.ast.FieldInstruction`:



### Public Member Functions

- [FieldInstruction](#) (final String a\_line\_text, final String a\_name)

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.*

- final boolean [correct](#) ()

*Field instruction line is correct if it has two parameters.*

- final Instruction [getInstruction](#) ()

*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for a field access instruction.*

### Static Public Member Functions

- static String[] [getMnemonics](#) ()

*This method returns the array of field [instructions](#) mnemonics.*

#### 6.48.1 Detailed Description

This class handles the creation and correctness for [instructions](#) to store and load field values.

These [instructions](#) are:

- getfield,
- getstatic,
- putfield,
- putstatic.

**Author:**

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

## 6.48.2 Constructor & Destructor Documentation

### 6.48.2.1 **umbra.instructions.ast.FieldInstruction.FieldInstruction (final String *a\_line\_text*, final String *a\_name*)**

This creates an instance of an instruction named as *a\_name* in a line the text of which is *a\_line\_text*. Currently it just calls the constructor of the superclass.

**Parameters:**

*a\_line\_text* the line text of the instruction

*a\_name* the mnemonic name of the instruction

**See also:**

[InstructionLineController](#).[InstructionLineController](#)(String, String)

## 6.48.3 Member Function Documentation

### 6.48.3.1 **static String [ ] umbra.instructions.ast.FieldInstruction.getMnemonics () [static]**

This method returns the array of field [instructions](#) mnemonics.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController](#).[getMnemonics](#)()

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

### 6.48.3.2 final boolean umbra.instructions.ast.FieldInstruction.correct ()

Field instruction line is correct if it has two parameters.

The first one is the name of the field and the second is the descriptor of the field. The precise format is: whitespace number : whitespace mnemonic whitespace fieldname typedescriptor whitespace ( whitespace number whitespace ) whitespace endline

**Returns:**

true when the syntax of the instruction line is correct

**See also:**

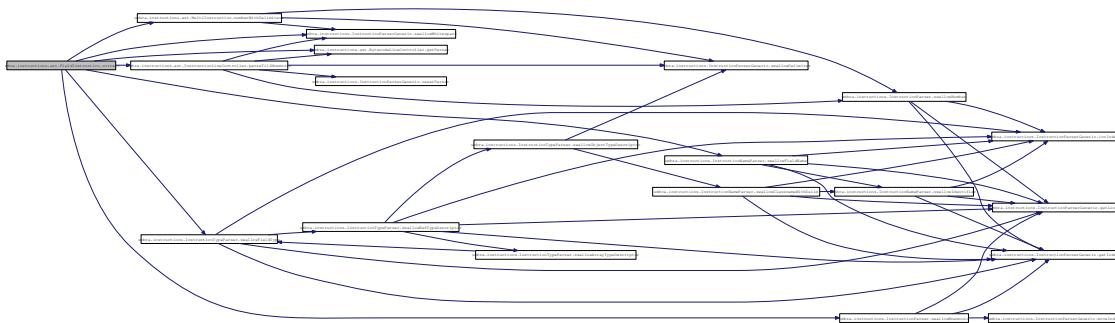
[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

References [umbra.instructions.ast.BytecodeLineController.getParser\(\)](#), [umbra.instructions.ast.MultiInstruction.numberWithDelimiters\(\)](#), [umbra.instructions.ast.InstructionLineController.parseTillMnemonic\(\)](#), [umbra.instructions.InstructionNameParser.swallowFieldName\(\)](#), [umbra.instructions.InstructionTypeParser.swallowFieldType\(\)](#), [umbra.instructions.InstructionParser.swallowMnemonic\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#)

Referenced by [umbra.instructions.ast.FieldInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



### 6.48.3.3 final Instruction umbra.instructions.ast.FieldInstruction.getInstruction ()

This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for a field access instruction.

It computes the index parameter of the instruction before the instruction is constructed. The method can construct one of the [instructions](#):

- getfield,
- getstatic,
- putfield,
- putstatic.

This method also checks the syntactical correctness of the current instruction line.

**Returns:**

the freshly constructed BCEL instruction or `null` in case the instruction is not a field access instruction and in case the instruction line is incorrect

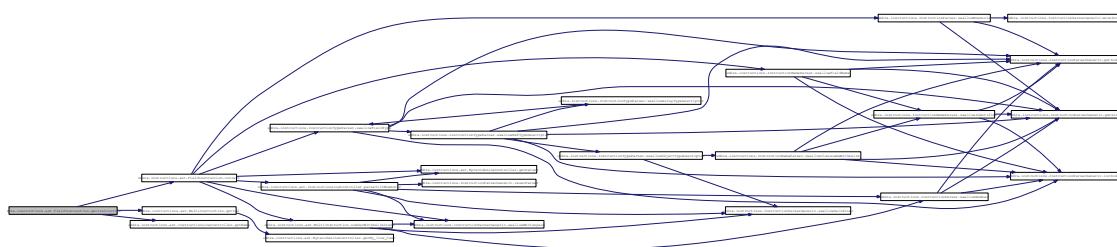
**See also:**

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References [umbra.instructions.ast.FieldInstruction.correct\(\)](#), [umbra.instructions.ast.MultiInstruction.getInd\(\)](#), and [umbra.instructions.ast.InstructionLineController.getName\(\)](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [source/umbra/instructions/ast/FieldInstruction.java](#)

## 6.49 umbra.lib.FileNames Class Reference

This is just container for operations on the file names used in the Umbra plugin (i.e.

### Static Public Member Functions

- static String [replaceLast](#) (final String a\_string, final String an\_old\_suffix, final String a\_new\_suffix)  
*This method replaces the last occurrence of the oldSuffix with the newSuffix in string.*
- static String [getClassPathSeparator](#) ()  
*This is a convenience method to obtain the classpath separator relevant to the current operating system.*
- static String [getFileSeparator](#) ()  
*This is a convenience method to obtain the separator that separates subsequent direcotry and file entries in a path to a resource.*
- static String [stripAllWhitespace](#) (final String a\_strip\_me)  
*This method strips off all the whitespace characters in the given string even inside the string.*
- static IFile [getClassFileFile](#) (final IFile a\_java\_file, final CompilationUnitEditor an\_editor) throws JavaModelException  
*This method gives the proper classfile file for a given Java file.*
- static IFile [getBTCFileName](#) (final IFile a\_file, final CompilationUnitEditor an\_editor)  
*This method gives the proper .btc file for a given Java file.*
- static String [getSavedClassFileNameForBTC](#) (final IPath a\_path)  
*This method returns for a given path to a .btc file a name of the classfile that was saved in order to keep the original copy of the classfile generated from the Java source code file.*
- static String [getSavedClassFileNameForPrefix](#) (final IPath a\_path, final String an\_extension)  
*This method returns for a given path to a file with the extension an\_extension a name of the class file that was saved in order to keep the original copy of the class file generated from the Java source code file.*
- static String [getSavedClassFileNameForClass](#) (final IPath a\_path)  
*This method returns for a given path to a .class file a name of the classfile that was saved in order to keep the original copy of the classfile generated from the Java source code file.*
- static IFile [getClassFileFileFor](#) (final IFile a\_java\_file, final AbstractTextEditor an\_editor, final String an\_extension) throws JavaModelException  
*This method gives the proper class file file for a given source file (usually Java or .btc file).*
- static int [getIndexAfter](#) (final String the\_data, final String a\_pattern)  
*The method finds the first occurrence of the pattern a\_pattern in the string the\_data and returns the index of the first character after the occurrence of the pattern.*
- static IPath [getClassFilePath](#) (final IType a\_java\_type) throws JavaModelException  
*This method returns java class file path file of e java element.*
- static IPath [getOutputTypePath](#) (final IType a\_java\_type) throws JavaModelException

*Method returns output path (containing output .class files) of the package where javaElement is situated.*

- static String [getPackageName](#) (final IJavaElement a\_java\_element)  
*This method returns a package of given IJavaElement.*
- static IJavaElement [getJavaElement](#) (final IEeditorPart an\_editor)  
*Method returns IJavaElement associated with IEeditorPart.*
- static IType [getEnclosingType](#) (final IJavaElement a\_java\_element)  
*Method returns enclosing IType for IJavaElement.*
- static IType [getSelectedType](#) (final IEeditorPart an\_editor) throws JavaModelException  
*Method returns the selected IType in IEeditorPart.*
- static IPath [getPath](#) (final IPath a\_path)  
*Transform a relative file path (inside the project) into the absolute one.*

## Static Public Attributes

- static final String **JAVA\_EXTENSION** = ".java"  
*The file extension for the Java files.*
- static final String **CLASS\_EXTENSION\_NONDOT** = "class"  
*The file extension for the Java class files, without dot.*
- static final String **BYTECODE\_HISTORY\_EXTENSION\_NONDOT** = "bt"  
*The file extension for the history files, without dot.*
- static final String **CLASS\_EXTENSION** = ".class"  
*The file extension for the Java class files.*
- static final String **BYTECODE\_EXTENSION** = ".btc"  
*The file extension for the files with the Umbra bytecode representation (i.e.*
- static final String **BYTECODE\_HISTORY\_EXTENSION** = ".bt"  
*The file extension for the history files.*
- static final boolean **DEBUG\_MODE** = true  
*This constant says if the debugging print outs should be generated.*

## Private Member Functions

- [FileNames](#) ()  
*A private empty constructor to prevent constructing of objects for this class.*

### 6.49.1 Detailed Description

This is just container for operations on the file names used in the Umbra plugin (i.e. `.java`, `.class`, `.btc`). It contains the methods to convert from one kind of name to another one with the whole logic.

FIXME: the logic should be as follows:

- the class files and all their historical versions should be kept where the output directory for the current project is
- the `.btc` files should be located where the `.java` files are  
<https://mobius.ucd.ie/ticket/546>

#### Author:

Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))  
Krzysztof Jakubczyk ([kjk@mimuw.edu.pl](mailto:kjk@mimuw.edu.pl))

#### Version:

a-01

### 6.49.2 Constructor & Destructor Documentation

#### 6.49.2.1 umbra.lib.FileNames.FileNames () [private]

A private empty constructor to prevent constructing of objects for this class.

### 6.49.3 Member Function Documentation

#### 6.49.3.1 static String umbra.lib.FileNames.replaceLast (final String *a\_string*, final String *an\_old\_suffix*, final String *a\_new\_suffix*) [static]

This method replaces the last occurrence of the `oldSuffix` with the `newSuffix` in `string`. It serves to exchange the file sufficies. In case `oldSuffix` does not occur in `string` it returns `string`.

#### Parameters:

`a_string` string to replace the suffix from  
`an_old_suffix` the suffix to replace  
`a_new_suffix` the new suffix

#### Returns:

the string with replaced suffix

Referenced by `umbra.lib.FileNames.getBTCFileName()`, `umbra.lib.FileNames.getClassFileFileFor()`, and `umbra.lib.FileNames.getSavedClassFileNameForPrefix()`.

**6.49.3.2 static String umbra.lib.FileNames.getClassPathSeparator () [static]**

This is a convenience method to obtain the classpath separator relevant to the current operating system.

**Returns:**

the string that separates the classpath entries

**6.49.3.3 static String umbra.lib.FileNames.getFileSeparator () [static]**

This is a convenience method to obtain the separator that separates subsequent directory and file entries in a path to a resource.

This value is relevant to the current operating system.

**Returns:**

the string that separates the file path entries

**6.49.3.4 static String umbra.lib.FileNames.stripAllWhitespace (final String *a\_strip\_me*) [static]**

This method strips off all the whitespace characters in the given string even inside the string.

**Parameters:**

*a\_strip\_me* the string to strip the whitespace from

**Returns:**

the string with the whitespace stripped off

**6.49.3.5 static IFile umbra.lib.FileNames.getClassFileFile (final IFile *a\_java\_file*, final CompilationUnitEditor *an\_editor*) throws JavaModelException [static]**

This method gives the proper classfile file for a given Java file.

**Parameters:**

*a\_java\_file* Java source code file for which we try to find the class file

*an\_editor* in which the .java file is edited

**Returns:**

the IFile for the corresponding .class file

**Exceptions:**

**JavaModelException** in case the project in which the *editor* operates has no class file output location set

References `umbra.lib.FileNames.getClassFileFileFor()`, and `umbra.lib.FileNames.JAVA_EXTENSION`.

Here is the call graph for this function:



#### 6.49.3.6 static IFile umbra.lib.FileNames.getBTCFileName (final IFile *a\_file*, final CompilationUnitEditor *an\_editor*) [static]

This method gives the proper .btc file for a given Java file.

**Parameters:**

*a\_file* Java source code file for which we try to find the .btc file  
*an\_editor* in which the `.java` file is edited

**Returns:**

the IFile for the corresponding .btc file

References `umbra.lib.FileNames.BYTECODE_EXTENSION`, `umbra.lib.FileNames.JAVA_EXTENSION`, and `umbra.lib.FileNames.replaceLast()`.

Here is the call graph for this function:



#### 6.49.3.7 static String umbra.lib.FileNames.getSavedClassNameForBTC (final IPath *a\_path*) [static]

This method returns for a given path to a .btc file a name of the classfile that was saved in order to keep the original copy of the classfile generated from the Java source code file.

No check is made that the path *a\_path* indeed has the extension.

**Parameters:**

*a\_path* a path to a .btc file

**Returns:**

corresponding name of the file with the saved version of the original bytecode

References `umbra.lib.FileNames.BYTECODE_EXTENSION`, and `umbra.lib.FileNames.getSavedClassNameForPrefix()`.

Here is the call graph for this function:



#### 6.49.3.8 static String umbra.lib.FileNames.getSavedClassNameForPrefix (final IPath *a\_path*, final String *an\_extension*) [static]

This method returns for a given path to a file with the extension *an\_extension* a name of the class file that was saved in order to keep the original copy of the class file generated from the Java source code file.

No check is made that the path indeed has the extension.

##### Parameters:

*a\_path* a path to a file

*an\_extension* the extension for which the original byte code file name is returned

##### Returns:

corresponding name of the file with the saved version of the original byte code

References umbra.lib.FileNames.CLASS\_EXTENSION, and umbra.lib.FileNames.replaceLast().

Referenced by umbra.lib.FileNames.getSavedClassNameForBTC(), and umbra.lib.FileNames.getSavedClassNameForClass().

Here is the call graph for this function:



#### 6.49.3.9 static String umbra.lib.FileNames.getSavedClassNameForClass (final IPath *a\_path*) [static]

This method returns for a given path to a .class file a name of the classfile that was saved in order to keep the original copy of the classfile generated from the Java source code file.

No check is made that the path indeed has the extension.

##### Parameters:

*a\_path* a path to a .class file

##### Returns:

corresponding name of the file with the saved version of the original bytecode

References umbra.lib.FileNames.CLASS\_EXTENSION, and umbra.lib.FileNames.getSavedClassNameForPrefix().

Here is the call graph for this function:



#### 6.49.3.10 static IFile umbra.lib.FileNames.getClassFileFor (final IFile *a\_java\_file*, final AbstractTextEditor *an\_editor*, final String *an\_extension*) throws JavaModelException [static]

This method gives the proper class file file for a given source file (usually Java or .btc file).

**Parameters:**

- a\_java\_file* a source code file for which we try to find the class file
- an\_editor* a Java file [editor](#) in which the corresponding Java file is edited
- an\_extension* an extension of the file for which we generate the .class file name (usually [.java](#) or [.btc](#))

**Returns:**

the [IFile](#) for the corresponding .class file

**Exceptions:**

- JavaModelException*** in case the project in which the [editor](#) operates has no class file output location set

References [umbra.lib.FileNames.CLASS\\_EXTENSION](#), and [umbra.lib.FileNames.replaceLast\(\)](#).

Referenced by [umbra.lib.FileNames.getClassFileFile\(\)](#).

Here is the call graph for this function:



#### 6.49.3.11 static int umbra.lib.FileNames.getIndexAfter (final String *the\_data*, final String *a\_pattern*) [static]

The method finds the first occurrence of the pattern *a\_pattern* in the string *the\_data* and returns the index of the first character after the occurrence of the pattern.

In case the pattern does not occur in the data string, the method returns a negative number.

**Parameters:**

- the\_data* the string in which we seek the index
- a\_pattern* a pattern we look for

**Returns:**

- the index of the first character after the first occurrence of the pattern; in case the pattern does not occur - a negative number

#### 6.49.3.12 static IPath umbra.lib.FileNames.getClassFilePath (final IType *a\_java\_type*) throws ***JavaModelException*** [static]

This method returns [java](#) class file path file of e [java](#) element.

Proposed usage: `getClassFilePath(getSelectedType(editor))`

**Parameters:**

- a\_java\_type* type to find output class file path

**Returns:**

output class file path

**Exceptions:**

*JavaModelException* if the output path for the current project does not exist

References      `umbra.lib.FileNames.CLASS_EXTENSION_NONDOT,`      and      `umbra.lib.FileNames.getOutputTypePath()`.

Here is the call graph for this function:



#### 6.49.3.13 static IPath umbra.lib.FileNames.getOutputTypePath (final IType *a\_java\_type*) throws *JavaModelException* [static]

Method returns output path (containing output .class files) of the package where javaElement is situated.

**Parameters:**

*a\_java\_type* the element to find output package of

**Returns:**

package output path of javaElement

**Exceptions:**

*JavaModelException* if the output path for the current project does not exist

Referenced by `umbra.lib.FileNames.getClassFilePath()`.

#### 6.49.3.14 static String umbra.lib.FileNames.getPackageName (final IJavaElement *a\_java\_element*) [static]

This method returns a package of given IJavaElement.

**Parameters:**

*a\_java\_element* the element we want to find package of

**Returns:**

`java` package name

#### 6.49.3.15 static IJavaElement umbra.lib.FileNames.getJavaElement (final IEeditorPart *an\_editor*) [static]

Method returns IJavaElement associated with IEeditorPart.

**Parameters:**

*an\_editor* the [editor](#) to get IJavaElement from

**Returns:**

[java](#) element associated with [editor](#)

Referenced by [umbra.lib.FileNames.getSelectedType\(\)](#).

**6.49.3.16 static IType umbra.lib.FileNames.getEnclosingType (final IJavaElement *a\_java\_element*) [static]**

Method returns enclosing IType for IJavaElement.

**Parameters:**

*a\_java\_element* the IJavaElement

**Returns:**

enclosing IType of javaElement

**6.49.3.17 static IType umbra.lib.FileNames.getSelectedType (final IEIPart *an\_editor*) throws JavaModelException [static]**

Method returns the selected IType in IEIPart.

**Parameters:**

*an\_editor* the [editor](#) to find IType. IMPORTANT: must be JavaEditor.

**Returns:**

IType selected in [editor](#)

**Exceptions:**

[JavaModelException](#) if the contents of the [editor](#) cannot be accessed

References [umbra.lib.FileNames.getJavaElement\(\)](#).

Here is the call graph for this function:

**6.49.3.18 static IPPath umbra.lib.FileNames.getPath (final IPPath *a\_path*) [static]**

Transform a relative file path (inside the project) into the absolute one.

**Parameters:**

*a\_path* a relative path

**Returns:**

the corresponding absolute path

#### 6.49.4 Member Data Documentation

##### 6.49.4.1 **final String umbra.lib.Filenames.JAVA\_EXTENSION = ".java" [static]**

The file extension for the Java files.

Referenced by `umbra.lib.Filenames.getBTCFileName()`, and `umbra.lib.Filenames.getClassFileFile()`.

##### 6.49.4.2 **final String umbra.lib.Filenames.CLASS\_EXTENSION\_NODOT = "class" [static]**

The file extension for the Java class files, without dot.

Referenced by `umbra.lib.Filenames.getClassFilePath()`.

##### 6.49.4.3 **final String umbra.lib.Filenames.BYTECODE\_HISTORY\_EXTENSION\_NODOT = "bt" [static]**

The file extension for the history files, without dot.

##### 6.49.4.4 **final String umbra.lib.Filenames.CLASS\_EXTENSION = ".class" [static]**

The file extension for the Java class files.

Referenced by `umbra.lib.Filenames.getClassFileFileFor()`, `umbra.lib.Filenames.getSavedClassNameForClass()`, and `umbra.lib.Filenames.getSavedClassNameForPrefix()`.

##### 6.49.4.5 **final String umbra.lib.Filenames.BYTECODE\_EXTENSION = ".btc" [static]**

The file extension for the files with the Umbra bytecode representation (i.e. `.btc`).

Referenced by `umbra.lib.Filenames.getBTCFileName()`, and `umbra.lib.Filenames.getSavedClassNameForBTC()`.

##### 6.49.4.6 **final String umbra.lib.Filenames.BYTECODE\_HISTORY\_EXTENSION = ".bt" [static]**

The file extension for the history files.

##### 6.49.4.7 **final boolean umbra.lib.Filenames.DEBUG\_MODE = true [static]**

This constant says if the debugging print outs should be generated.

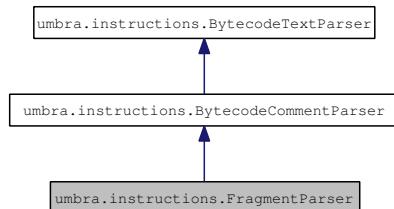
The documentation for this class was generated from the following file:

- source/umbra/lib/[Filenames.java](#)

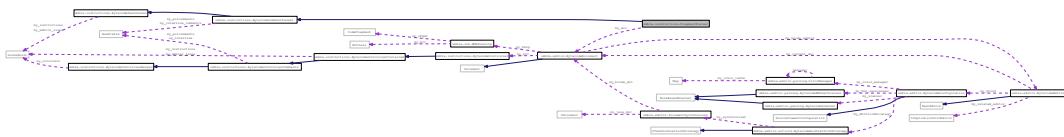
## 6.50 umbra.instructions.FragmentParser Class Reference

This class is used to parse fragments of byte code textual documents.

Inheritance diagram for umbra.instructions.FragmentParser:



Collaboration diagram for umbra.instructions.FragmentParser:



### Private Attributes

- final [BytecodeDocument my\\_doc](#)

*The document which contains the fragment to be parsed.*

- final int [my\\_start](#)

*The first line to be parsed.*

- final int [my\\_end](#)

*The last line to be parsed.*

### 6.50.1 Detailed Description

This class is used to parse fragments of byte code textual documents.

Currently it handles only fragments included in a single method.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

## 6.50.2 Member Data Documentation

### 6.50.2.1 final BytecodeDocument umbra.instructions.FragmentParser.my\_doc [private]

The document which contains the fragment to be parsed.

### 6.50.2.2 final int umbra.instructions.FragmentParser.my\_start [private]

The first line to be parsed.

The parsing includes this line.

### 6.50.2.3 final int umbra.instructions.FragmentParser.my\_end [private]

The last line to be parsed.

The parsing includes this line.

The documentation for this class was generated from the following file:

- source/umbra/instructions/[FragmentParser.java](#)

## 6.51 umbra.lib.GUIMessages Class Reference

This is just container for texts of all the GUI messages.

### Static Public Member Functions

- static String [substitute](#) (final String a\_message, final String a\_substitute)  
*This method substitutes in the given message all the template points with the given substitute string.*
- static String [substitute2](#) (final String a\_message, final String a\_substitute, final String a\_substitute2)  
*This method substitutes in the given message all the template points with the given substitute string.*
- static void [exceededRangeInfo](#) (final Shell a\_shell, final [UmbraRangeException](#) an\_ex, final String a\_title)  
*This method displays error message for [UmbraRangeException](#) signals.*
- static void [wrongClassFileOptMessage](#) (final Shell a\_shell, final String a\_title)  
*Displays the message that the current project has no output path for Java class files.*

### Static Public Attributes

- static final String [SUBSTITUTE](#) = "#1#"  
*A string to indicate a point in a string template where the data to instantiate the template should be substituted.*
- static final String [SUBSTITUTE2](#) = "#2#"  
*A string to indicate a point in a string template where the second portion of data to instantiate the template should be substituted.*
- static final String [BYTECODE\\_MESSAGE\\_TITLE](#) = "Bytecode"  
*A string used as a generic header in the message panes launched in connection with the byte code text editor.*
- static final String [DISAS\\_MESSAGE\\_TITLE](#)  
*A string used as a header in the message panes launched in connection with the Java source code action to disassemble code (class [umbra.java.actions.DisasBCEL](#)).*
- static final String [SYNCH\\_MESSAGE\\_TITLE](#)  
*A string used as a header in the message panes launched in connection with the actions to synchronise the code (classes [umbra.java.actions.SynchrSBAction](#) and [umbra.editor.actions.BytecodeSynchrAction](#)).*
- static final String [COMMIT\\_MESSAGE\\_TITLE](#)  
*A string used as a header in the message panes launched in connection with the Java source code action to commit changes (class [umbra.java.actions.CommitAction](#)).*
- static final String [INVALID\\_EDIT\\_OPERATION](#) = "Invalid edit operation"  
*The message which informs the user that the operation he/she wants to carry out cannot be performed.*

- static final String **DISAS\_SAVE\_BYTECODE\_FIRST**

*The message which requires the user to save the byte code before it is disassembled.*

- static final String **DISAS\_SAVING\_PROBLEMS**

*The message which informs the user that the file cannot be saved under the given location.*

- static final String **DISAS\_LOADING\_PROBLEMS**

*The message which informs the user that the file cannot be saved under the given location.*

- static final String **FILED\_CLASS\_FILE\_OPERATION**

*The message which informs the user that an operation on a class file failed.*

- static final String **DISAS\_CLASSFILEOUTPUT\_PROBLEMS**

*The message which informs that the current project has no class file output location set.*

- static final String **DISAS\_PATH\_DOES\_NOT\_EXIST**

*The message which informs the user that a path does not exists.*

- static final String **DISAS\_EDITOR\_PROBLEMS**

*The message which informs the user that the **editor** cannot be created or initialised.*

- static final String **NO\_LINE\_IN\_DOC**

*The message which informs the user that the document does not contain a line of the given number.*

- static final String **NO\_POSITION\_IN\_DOC**

*The message which informs the user that the document does not contain a position of the given number.*

- static final String **NO\_METHODS\_IN\_DOC**

*The message which informs the user that the document does not contain a method of the given number.*

- static final String **INVALID\_EXTENSION**

*A template message that warns about wrong file type.*

- static final String **WRONG\_LOCATION\_MSG**

*A template message that warns about wrong location in a document.*

- static final String **WRONG\_SYNCHRONISATION\_MSG**

*The message informs the user that the synchronisation is impossible.*

- static final String **WRONG\_JAVAACCESS\_MSG**

*The message informs the user that access to a Java element is impossible.*

- static final String **NOINSTRUCTION\_MSG**

*The message informs the user that the position cannot be associated with an instruction in a reasonable way.*

## Private Member Functions

- **GUIMessages ()**

*The empty constructor to forbid the creation of the instances.*

### 6.51.1 Detailed Description

This is just container for texts of all the GUI messages.

FIXME: this does not contain all the messages <https://mobius.ucd.ie/ticket/591>

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.51.2 Constructor & Destructor Documentation

#### 6.51.2.1 **umbra.lib.GUIMessages.GUIMessages () [private]**

The empty constructor to forbid the creation of the instances.

### 6.51.3 Member Function Documentation

#### 6.51.3.1 **static String umbra.lib.GUIMessages.substitute (final String *a\_message*, final String *a\_substitute*) [static]**

This method substitutes in the given message all the template points with the given substitute string.

##### Parameters:

*a\_message* a message to substitute template positions

*a\_substitute* a string to fill in the template positions

##### Returns:

a string with the template positions properly substituted

References `umbra.lib.GUIMessages.SUBSTITUTE`.

#### 6.51.3.2 **static String umbra.lib.GUIMessages.substitute2 (final String *a\_message*, final String *a\_substitute*, final String *a\_substitute2*) [static]**

This method substitutes in the given message all the template points with the given substitute string.

##### Parameters:

*a\_message* a message to substitute template positions

*a\_substitute* a string to fill in the first kind of the template positions  
*a\_substitute2* a string to fill in the second kind of the template positions

**Returns:**

a string with the template positions properly substituted

References `umbra.lib.GUIMessages.SUBSTITUTE`, and `umbra.lib.GUIMessages.SUBSTITUTE2`.

#### 6.51.3.3 static void `umbra.lib.GUIMessages.exceededRangeInfo` (final Shell *a\_shell*, final UmbraRangeException *an\_ex*, final String *a\_title*) [static]

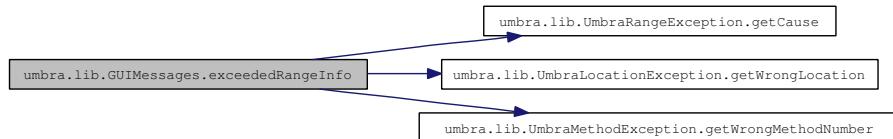
This method displays error message for [UmbraRangeException](#) signals.

**Parameters:**

*a\_shell* a shell which displays the messages  
*an\_ex* an exception which caused the need of the message  
*a\_title* a title of the message window

References `umbra.lib.UmbraRangeException.getCause()`, `umbra.lib.UmbraLocationException.getWrongLocation()`, `umbra.lib.UmbraMethodException.getWrongMethodNumber()`, `umbra.lib.GUIMessages.NO_LINE_IN_DOC`, `umbra.lib.GUIMessages.NO_METHODS_IN_DOC`, and `umbra.lib.GUIMessages.SUBSTITUTE`.

Here is the call graph for this function:



#### 6.51.3.4 static void `umbra.lib.GUIMessages.wrongClassFileOptMessage` (final Shell *a\_shell*, final String *a\_title*) [static]

Displays the message that the current project has no output path for Java class files.

**Parameters:**

*a\_shell* the shell which displays the message  
*a\_title* the title of the message window

References `umbra.lib.GUIMessages.DISAS_CLASSFILEOUTPUT_PROBLEMS`.

### 6.51.4 Member Data Documentation

#### 6.51.4.1 final String `umbra.lib.GUIMessages.SUBSTITUTE = "#1#"` [static]

A string to indicate a point in a string template where the data to instantiate the template should be substituted.

Referenced by `umbra.lib.GUIMessages.exceededRangeInfo()`, `umbra.lib.GUIMessages.substitute()`, and `umbra.lib.GUIMessages.substitute2()`.

**6.51.4.2 final String umbra.lib.GUIMessages.SUBSTITUTE2 = "#2#" [static]**

A string to indicate a point in a string template where the second portion of data to instantiate the template should be substituted.

Referenced by `umbra.lib.GUIMessages.substitute2()`.

**6.51.4.3 final String umbra.lib.GUIMessages.BYTECODE\_MESSAGE\_TITLE = "Bytecode" [static]**

A string used as a generic header in the message panes launched in connection with the byte code text editor.

**6.51.4.4 final String umbra.lib.GUIMessages.DISAS\_MESSAGE\_TITLE [static]****Initial value:**

`"Disassemble Bytecode"`

A string used as a header in the message panes launched in connection with the Java source code action to disassemble code (class [umbra.java.actions.DisasBCEL](#)).

**6.51.4.5 final String umbra.lib.GUIMessages.SYNCH\_MESSAGE\_TITLE [static]****Initial value:**

`"Synchronisation"`

A string used as a header in the message panes launched in connection with the actions to synchronise the code (classes [umbra.java.actions.SynchrSBAction](#) and [umbra.editor.actions.BytecodeSynchrAction](#)).

**6.51.4.6 final String umbra.lib.GUIMessages.COMMIT\_MESSAGE\_TITLE [static]****Initial value:**

`"Committing changes"`

A string used as a header in the message panes launched in connection with the Java source code action to commit changes (class [umbra.java.actions.CommitAction](#)).

**6.51.4.7 final String umbra.lib.GUIMessages.INVALID\_EDIT\_OPERATION = "Invalid edit operation" [static]**

The message which informs the user that the operation he/she wants to carry out cannot be performed.

**6.51.4.8 final String umbra.lib.GUIMessages.DISAS\_SAVE\_BYTECODE\_FIRST [static]****Initial value:**

`"You must save the source code before you can show its byte code."`

The message which requires the user to save the byte code before it is disassembled.

**6.51.4.9 final String umbra.lib.GUIMessages.DISAS\_SAVING\_PROBLEMS [static]****Initial value:**

```
"Problems with saving the file under the given location"
```

The message which informs the user that the file cannot be saved under the given location.

**6.51.4.10 final String umbra.lib.GUIMessages.DISAS\_LOADING\_PROBLEMS [static]****Initial value:**

```
"Problems with loading the file under the given location: "
```

The message which informs the user that the file cannot be saved under the given location.

**6.51.4.11 final String umbra.lib.GUIMessages.FILED\_CLASS\_FILE\_OPERATION [static]****Initial value:**

```
"A file operation on the class file failed"
```

The message which informs the user that an operation on a class file failed.

**6.51.4.12 final String umbra.lib.GUIMessages.DISAS\_CLASSFILEOUTPUT\_PROBLEMS [static]****Initial value:**

```
"The current project has no class file output location set"
```

The message which informs that the current project has no class file output location set.

Referenced by `umbra.lib.GUIMessages.wrongClassFileOptMessage()`.

**6.51.4.13 final String umbra.lib.GUIMessages.DISAS\_PATH\_DOES\_NOT\_EXIST [static]****Initial value:**

```
"The path does not exist"
```

The message which informs the user that a path does not exists.

**6.51.4.14 final String umbra.lib.GUIMessages.DISAS\_EDITOR\_PROBLEMS [static]****Initial value:**

```
"The byte code editor cannot be opened or initialised"
```

The message which informs the user that the [editor](#) cannot be created or initialised.

**6.51.4.15 final String umbra.lib.GUIMessages.NO\_LINE\_IN\_DOC [static]****Initial value:**

```
"The current document has no positions for the line "
```

The message which informs the user that the document does not contain a line of the given number.

Referenced by `umbra.lib.GUIMessages.exceededRangeInfo()`.

**6.51.4.16 final String umbra.lib.GUIMessages.NO\_POSITION\_IN\_DOC [static]****Initial value:**

```
"The current document has no position "
```

The message which informs the user that the document does not contain a position of the given number.

**6.51.4.17 final String umbra.lib.GUIMessages.NO\_METHODS\_IN\_DOC [static]****Initial value:**

```
"The current document has too many methods (" + SUBSTITUTE + ")"
```

The message which informs the user that the document does not contain a method of the given number.

Referenced by `umbra.lib.GUIMessages.exceededRangeInfo()`.

**6.51.4.18 final String umbra.lib.GUIMessages.INVALID\_EXTENSION [static]****Initial value:**

```
"This is not a \" + SUBSTITUTE + "\" file."
```

A template message that warns about wrong file type.

**6.51.4.19 final String umbra.lib.GUIMessages.WRONG\_LOCATION\_MSG [static]****Initial value:**

```
"Wrong location " + SUBSTITUTE + " in a " + SUBSTITUTE2 + " document."
```

A template message that warns about wrong location in a document.

**6.51.4.20 final String umbra.lib.GUIMessages.WRONG\_SYNCHRONISATION\_MSG [static]****Initial value:**

```
"This position cannot be synchronised."
```

The message informs the user that the synchronisation is impossible.

**6.51.4.21 final String umbra.lib.GUIMessages.WRONG\_JAVAACCESS\_MSG [static]****Initial value:**

```
"A Java element cannot be accessed."
```

The message informs the user that access to a Java element is impossible.

**6.51.4.22 final String umbra.lib.GUIMessages.NOINSTRUCTION\_MSG [static]****Initial value:**

```
"No source code instruction can be associated with the given position"
```

The message informs the user that the position cannot be associated with an instruction in a reasonable way.

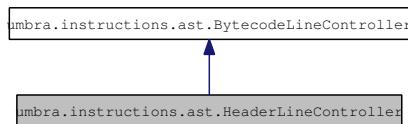
The documentation for this class was generated from the following file:

- source/umbra/lib/[GUIMessages.java](#)

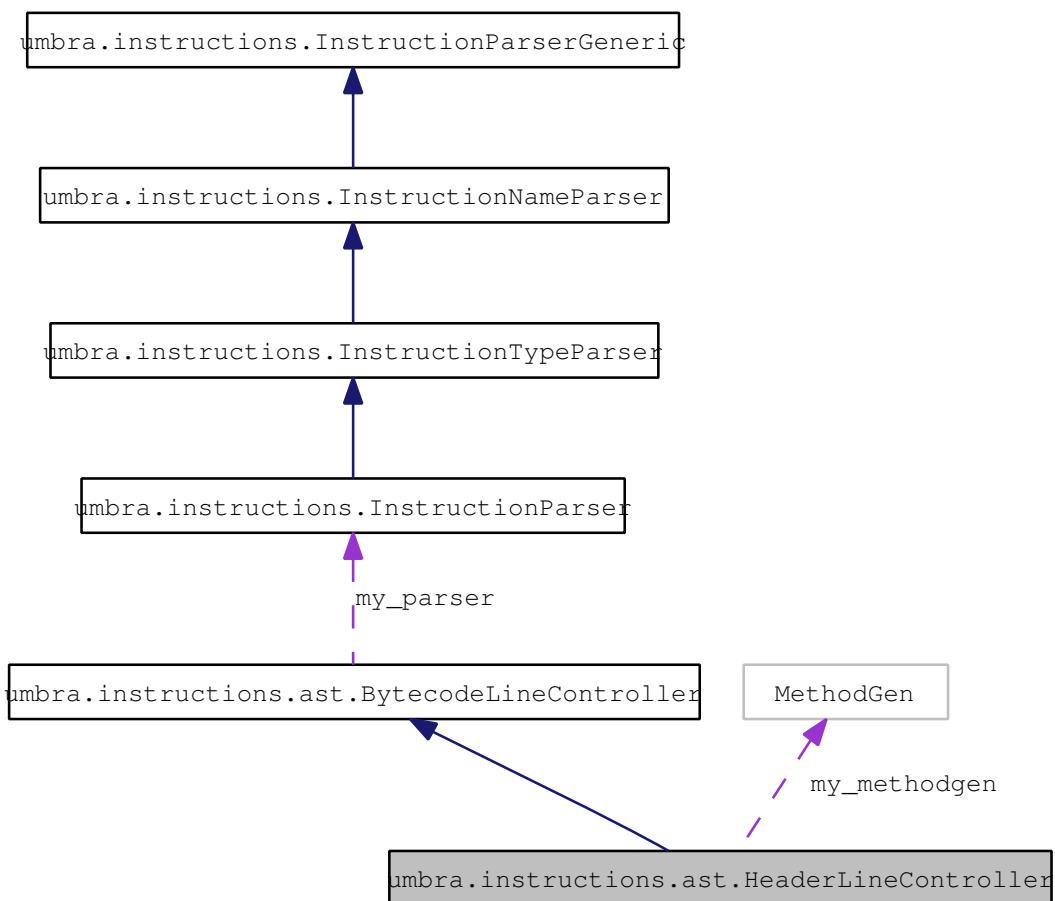
## 6.52 **umbra.instructions.ast.HeaderLineController** Class Reference

This is a class for lines in bytecode files that occur at the beginning of methods.

Inheritance diagram for **umbra.instructions.ast.HeaderLineController**:



Collaboration diagram for **umbra.instructions.ast.HeaderLineController**:



### Public Member Functions

- **HeaderLineController** (final String a\_line\_text)

*This creates an instance of a bytecode line handle which occurs at the beginning of a method a\_line.*

- final boolean **correct** ()

*Checks the correctness of the current header line.*

- final MethodGen [getMethod\(\)](#)  
*Returns the [MethodGen](#) structure responsible for the method in which the instruction resides.*
- final void [setMethod](#)(final MethodGen a\_mg)  
*Sets the [MethodGen](#) structure responsible for the method the header of which resides in the current object.*

## Private Attributes

- MethodGen [my\\_methodgen](#)  
*A [BCEL](#) object that represents the method the header of which is in the current object.*

### 6.52.1 Detailed Description

This is a class for lines in bytecode files that occur at the beginning of methods.

These are intended not to be edited by a user.

#### Author:

Tomek Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

### 6.52.2 Constructor & Destructor Documentation

#### 6.52.2.1 [umbra.instructions.ast.HeaderLineController.HeaderLineController](#) (final String *a\_line\_text*)

This creates an instance of a bytecode line handle which occurs at the beginning of a method *a\_line*.

Currently it just calls the constructor of the superclass.

#### Parameters:

*a\_line\_text* the string representation of the line text

#### See also:

[BytecodeLineController](#).[BytecodeLineController](#)(String)

### 6.52.3 Member Function Documentation

#### 6.52.3.1 final boolean [umbra.instructions.ast.HeaderLineController.correct\(\)](#)

Checks the correctness of the current header line.

This method checks only the approximate format. It checks if the header line starts with one of the fixed prefixes. The prefixes are enumerated in [BytecodeStrings#HEADER\\_PREFIX](#).

**Returns:**

`true` when the line starts with a header prefix, `false` otherwise

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References [umbra.instructions.ast.BytecodeLineController.getMy\\_line\\_text\(\)](#).

Here is the call graph for this function:

**6.52.3.2 final MethodGen umbra.instructions.ast.HeaderLineController.getMethod ()**

Returns the [MethodGen](#) structure responsible for the method in which the instruction resides.

**Returns:**

the method in which the current instruction is located

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References [umbra.instructions.ast.HeaderLineController.my\\_methodgen](#).

**6.52.3.3 final void umbra.instructions.ast.HeaderLineController.setMethod (final MethodGen a\_mg)**

Sets the [MethodGen](#) structure responsible for the method the header of which resides in the current object.

**Parameters:**

`a_mg` the [MethodGen](#) structure to associate with the header

References [umbra.instructions.ast.HeaderLineController.my\\_methodgen](#).

**6.52.4 Member Data Documentation****6.52.4.1 MethodGen umbra.instructions.ast.HeaderLineController.my\_methodgen [private]**

A BCEL object that represents the method the header of which is in the current object.

Referenced by [umbra.instructions.ast.HeaderLineController.getMethod\(\)](#), and [umbra.instructions.ast.HeaderLineController.setMethod\(\)](#).

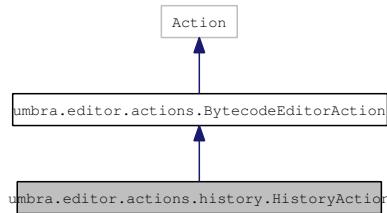
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[HeaderLineController.java](#)

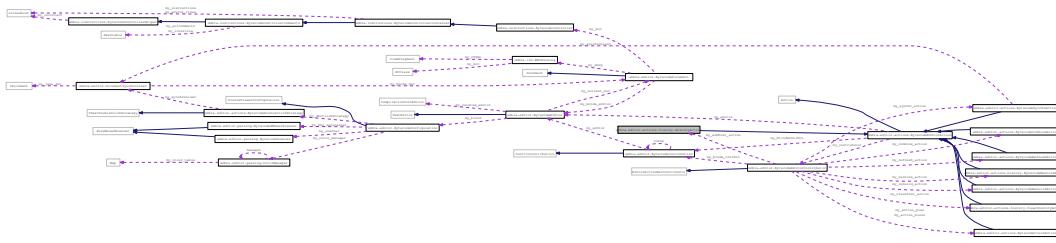
## 6.53 umbra.editor.actions.history.HistoryAction Class Reference

This class defines an action that adds current byte code snapshot to the [history](#) stack.

Inheritance diagram for `umbra.editor.actions.history.HistoryAction`:



Collaboration diagram for `umbra.editor.actions.history.HistoryAction`:



### Public Member Functions

- `HistoryAction (final BytecodeEditorContributor a_contributor, final BytecodeContribution a_btcd_-contribution)`

*This constructor creates the action to add item to the [history](#) of the byte code [editor](#).*

- `final void run ()`

*This method increments the counter of the existing [history](#) snapshots.*

- `void selectionChanged (final IAction an_action, final ISelection a_selection)`

*The method reacts when the selected area changes in the bytecode [editor](#).*

### 6.53.1 Detailed Description

This class defines an action that adds current byte code snapshot to the [history](#) stack.

The stack is implemented in the file system.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

## 6.53.2 Constructor & Destructor Documentation

### 6.53.2.1 umbra.editor.actions.history.HistoryAction.HistoryAction (final BytecodeEditorContributor *a\_contributor*, final BytecodeContribution *a\_btcd\_contribution*)

This constructor creates the action to add item to the [history](#) of the byte code [editor](#).

It registers the name of the action with the text "Add to history" and stores locally the object which creates all the [actions](#) and which contributes the [editor](#) GUI elements to the eclipse GUI.

**Parameters:**

*a\_contributor* the manager that initialises all the [actions](#) within the bytecode plugin

*a\_btcd\_contribution* the GUI elements contributed to the eclipse GUI by the bytecode [editor](#). This reference should be the same as in the parameter *a\_contributor*.

## 6.53.3 Member Function Documentation

### 6.53.3.1 final void umbra.editor.actions.history.HistoryAction.run ()

This method increments the counter of the existing [history](#) snapshots.

In case the [history](#) stack is full an appropriate message is displayed. Otherwise, the files for the currently edited bytecode file (i.e. .btc file and .class file) are saved into the [history](#).

References [umbra.editor.actions.BytecodeEditorAction.getEditor\(\)](#).

Here is the call graph for this function:



### 6.53.3.2 void umbra.editor.actions.history.HistoryAction.selectionChanged (final IAction *an\_action*, final ISelection *a\_selection*)

The method reacts when the selected area changes in the bytecode [editor](#).

Currently, it does nothing.

**Parameters:**

*an\_action* the action which triggered the selection change

*a\_selection* the new selection.

The documentation for this class was generated from the following file:

- source/umbra/editor/actions/history/[HistoryAction.java](#)

## 6.54 umbra.lib.HistoryOperations Class Reference

This class implements the operations on history items.

### Static Public Member Functions

- static void [saveBTCHistoryFile](#) (final IFile a\_file\_from, final int a\_hist\_num, final CompilationUnitEditor an\_editor) throws CoreException

*This method saves under the history slot number in a\_hist\_num the bytecode classfile that corresponds to the file in a\_file\_from.*

- static void [saveClassHistoryFile](#) (final IFile a\_file\_from, final int a\_hist\_num, final CompilationUnitEditor an\_editor) throws CoreException

*This method saves under the history slot number in a\_hist\_num the bytecode classfile that corresponds to the file in a\_file\_from.*

- static void [loadBTCHistoryFile](#) (final IFile a\_file\_from, final int a\_hist\_num, final CompilationUnitEditor an\_editor) throws CoreException

*This method loads from the history slot number in a\_hist\_num the .btc file that corresponds to the file in a\_file\_from.*

- static void [loadClassHistoryFile](#) (final IFile a\_file\_from, final int a\_hist\_num, final CompilationUnitEditor an\_editor) throws CoreException

*This method loads from the history slot number in a\_hist\_num the class file that corresponds to the .btc file in a\_file\_from.*

- static void [removeBTCHistoryFile](#) (final IFile a\_file\_from, final int a\_hist\_num, final CompilationUnitEditor an\_editor) throws CoreException

*This method removes from the history slot number in a\_hist\_num the .btc file that corresponds to the file in a\_file\_from.*

- static void [removeClassHistoryFile](#) (final IFile a\_file\_from, final int a\_hist\_num, final CompilationUnitEditor an\_editor) throws CoreException

*This method removes from the history slot number in a\_hist\_num the class file that corresponds to the .btc file in a\_file\_from.*

### Static Public Attributes

- static final int [MAX\\_HISTORY](#) = 2

*The maximal number of history snapshots.*

- static final int [MIN\\_HISTORY](#) = 0

*The minimal number of history snapshots.*

- static final int [DEFAULT\\_HISTORY](#) = 0

*The default value of the history number, used in case none is given or in case an invalid number is used.*

## Private Member Functions

- [HistoryOperations \(\)](#)

*A private empty constructor to prevent constructing of objects for this class.*

## Static Private Member Functions

- static IFile [getHistoryBTCFile](#) (final IFile a\_file\_from, final int a\_hist\_num)  
*Obtains the historical version of the given .btc [IFile](#).*
- static IFile [getHistoryFile](#) (final IFile a\_file\_from, final int a\_hist\_num, final String an\_ext)  
*Obtains the historical version of a file with the given extension for the given .btc [IFile](#).*
- static IFile [getHistoryClassFile](#) (final IFile a\_file\_from, final int a\_hist\_num)  
*Obtains the historical version of the class file for the given .btc [IFile](#).*

### 6.54.1 Detailed Description

This class implements the operations on history items.

It implements the operations to save and load historical versions of .btc files and .class files.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.54.2 Constructor & Destructor Documentation

#### 6.54.2.1 umbra.lib.HistoryOperations.HistoryOperations () [private]

A private empty constructor to prevent constructing of objects for this class.

### 6.54.3 Member Function Documentation

#### 6.54.3.1 static void umbra.lib.HistoryOperations.saveBTCHistoryFile (final IFile a\_file\_from, final int a\_hist\_num, final CompilationUnitEditor an\_editor) throws CoreException [static]

This method saves under the history slot number in a\_hist\_num the bytecode classfile that corresponds to the file in a\_file\_from.

The [editor](#) is given to make the interface compatible with [saveClassHistoryFile\(IFile, int, CompilationUnitEditor\)](#).

#### Parameters:

*a\_file\_from* a .btc file for which the class file is to be inserted into the history

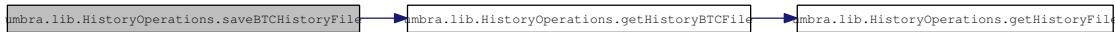
*a\_hist\_num* a history slot number under which the file should be saved  
*an\_editor* editor which edits the Java file corresponding to the Java file

#### Exceptions:

*CoreException* in case the file system operations cannot be performed

References `umbra.lib.HistoryOperations.getHistoryBTCFile()`.

Here is the call graph for this function:



#### 6.54.3.2 static IFile umbra.lib.HistoryOperations.getHistoryBTCFile (final IFile *a\_file\_from*, final int *a\_hist\_num*) [static, private]

Obtains the historical version of the given .btc **IFile**.

#### Parameters:

*a\_file\_from* a .btc file to retrieve the historical version for  
*a\_hist\_num* the number of the item to retrieve from the history

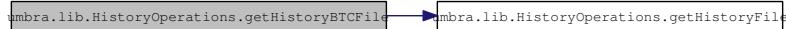
#### Returns:

the historical version of the file

References `umbra.lib.HistoryOperations.getHistoryFile()`.

Referenced by `umbra.lib.HistoryOperations.loadBTCHistoryFile()`, `umbra.lib.HistoryOperations.removeBTCHistoryFile()`, and `umbra.lib.HistoryOperations.saveBTCHistoryFile()`.

Here is the call graph for this function:



#### 6.54.3.3 static IFile umbra.lib.HistoryOperations.getHistoryFile (final IFile *a\_file\_from*, final int *a\_hist\_num*, final String *an\_ext*) [static, private]

Obtains the historical version of a file with the given extension for the given .btc **IFile**.

It removes the extension from the given .btc file and replaces it with the given extension concatenated with the historical item number.

#### Parameters:

*a\_file\_from* a .btc file to retrieve the historical version for  
*a\_hist\_num* the number of the item to retrieve from the history  
*an\_ext* the extension of the resulting file

**Returns:**

the historical version of the file with the given extension

Referenced by umbra.lib.HistoryOperations.getHistoryBTCFile(), and umbra.lib.HistoryOperations.getHistoryClassFile().

**6.54.3.4 static void umbra.lib.HistoryOperations.saveClassHistoryFile (final IFile *a\_file\_from*, final int *a\_hist\_num*, final CompilationUnitEditor *an\_editor*) throws CoreException [static]**

This method saves under the history slot number in *a\_hist\_num* the bytecode classfile that corresponds to the file in *a\_file\_from*.

**Parameters:**

*a\_file\_from* a .btc file for which the classfile is to be inserted into the history

*a\_hist\_num* a history slot number under which the file should be saved

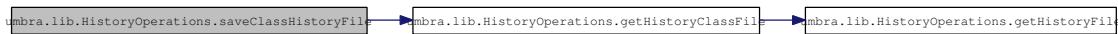
*an\_editor* editor which edits the Java file corresponding to the Java file

**Exceptions:**

*CoreException* in case the file system operations cannot be performed

References umbra.lib.HistoryOperations.getHistoryClassFile().

Here is the call graph for this function:



**6.54.3.5 static IFile umbra.lib.HistoryOperations.getHistoryClassFile (final IFile *a\_file\_from*, final int *a\_hist\_num*) [static, private]**

Obtains the historical version of the class file for the given .btc **IFile**.

**Parameters:**

*a\_file\_from* a .btc file to retrieve the historical version for

*a\_hist\_num* the number of the item to retrieve from the history

**Returns:**

the historical version of the file

References umbra.lib.HistoryOperations.getHistoryFile().

Referenced by umbra.lib.HistoryOperations.loadClassHistoryFile(), umbra.lib.HistoryOperations.removeClassHistoryFile(), and umbra.lib.HistoryOperations.saveClassHistoryFile().

Here is the call graph for this function:



**6.54.3.6 static void umbra.lib.HistoryOperations.loadBTCHistoryFile (final IFile *a\_file\_from*, final int *a\_hist\_num*, final CompilationUnitEditor *an\_editor*) throws CoreException [static]**

This method loads from the history slot number in *a\_hist\_num* the .btc file that corresponds to the file in *a\_file\_from*.

The *editor* is given to make the interface compatible with `saveClassHistoryFile(IFile, int, CompilationUnitEditor)`.

#### Parameters:

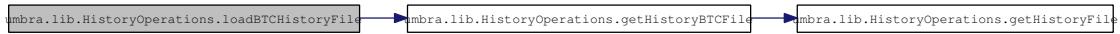
*a\_file\_from* a .btc file for which the .btc file is to be loaded from the history  
*a\_hist\_num* a history slot number from which the file should be loaded  
*an\_editor* *editor* which edits the Java file corresponding to the Java file

#### Exceptions:

*CoreException* in case the file system operations cannot be performed

References `umbra.lib.HistoryOperations.getHistoryBTCFile()`.

Here is the call graph for this function:



**6.54.3.7 static void umbra.lib.HistoryOperations.loadClassHistoryFile (final IFile *a\_file\_from*, final int *a\_hist\_num*, final CompilationUnitEditor *an\_editor*) throws CoreException [static]**

This method loads from the history slot number in *a\_hist\_num* the class file that corresponds to the .btc file in *a\_file\_from*.

The *editor* is given to make the interface compatible with `saveClassHistoryFile(IFile, int, CompilationUnitEditor)`.

#### Parameters:

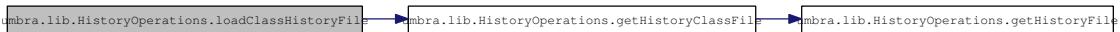
*a\_file\_from* a .btc file for which the class file is to be loaded from the history  
*a\_hist\_num* a history slot number from which the file should be loaded  
*an\_editor* *editor* which edits the Java file corresponding to the Java file

#### Exceptions:

*CoreException* in case the file system operations cannot be performed

References `umbra.lib.HistoryOperations.getHistoryClassFile()`.

Here is the call graph for this function:



**6.54.3.8 static void umbra.lib.HistoryOperations.removeBTCHistoryFile (final IFile *a\_file\_from*, final int *a\_hist\_num*, final CompilationUnitEditor *an\_editor*) throws CoreException [static]**

This method removes from the history slot number in *a\_hist\_num* the .btc file that corresponds to the file in *a\_file\_from*.

The *editor* is given to make the interface compatible with [saveClassHistoryFile\(IFile, int, CompilationUnitEditor\)](#).

#### Parameters:

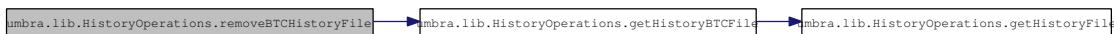
*a\_file\_from* a .btc file for which the .btc file is to be removed from the history  
*a\_hist\_num* a history slot number from which the file should be removed  
*an\_editor* *editor* which edits the Java file corresponding to the Java file

#### Exceptions:

*CoreException* in case the file system operations cannot be performed

References [umbra.lib.HistoryOperations.getHistoryBTCFile\(\)](#).

Here is the call graph for this function:



**6.54.3.9 static void umbra.lib.HistoryOperations.removeClassHistoryFile (final IFile *a\_file\_from*, final int *a\_hist\_num*, final CompilationUnitEditor *an\_editor*) throws CoreException [static]**

This method removes from the history slot number in *a\_hist\_num* the class file that corresponds to the .btc file in *a\_file\_from*.

The *editor* is given to make the interface compatible with [saveClassHistoryFile\(IFile, int, CompilationUnitEditor\)](#).

#### Parameters:

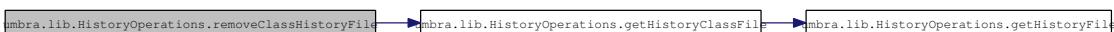
*a\_file\_from* a .btc file for which the class file is to be removed from the history  
*a\_hist\_num* a history slot number from which the file should be removed  
*an\_editor* *editor* which edits the Java file corresponding to the Java file

#### Exceptions:

*CoreException* in case the file system operations cannot be performed

References [umbra.lib.HistoryOperations.getHistoryClassFile\(\)](#).

Here is the call graph for this function:



## 6.54.4 Member Data Documentation

### 6.54.4.1 final int umbra.lib.HistoryOperations.MAX\_HISTORY = 2 [static]

The maximal number of history snapshots.

### 6.54.4.2 final int umbra.lib.HistoryOperations.MIN\_HISTORY = 0 [static]

The minimal number of history snapshots.

### 6.54.4.3 final int umbra.lib.HistoryOperations.DEFAULT\_HISTORY = 0 [static]

The default value of the history number, used in case none is given or in case an invalid number is used.

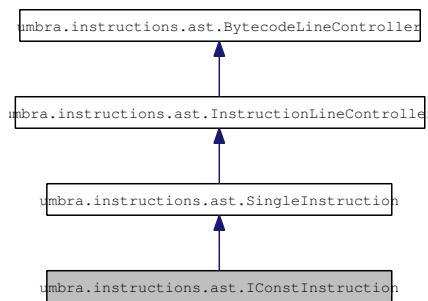
The documentation for this class was generated from the following file:

- source/umbra/lib/[HistoryOperations.java](#)

## 6.55 umbra.instructions.ast.IConstInstruction Class Reference

This class handles the creation and correctness for iconst [instructions](#) with no parameters.

Inheritance diagram for umbra.instructions.ast.IConstInstruction:



Collaboration diagram for umbra.instructions.ast.IConstInstruction:



### Public Member Functions

- [IConstInstruction](#) (final String a\_line\_text, final String a\_name)

*This creates an instance of an instruction named as a\_name with the line text a\_line.*

- Instruction [getInstruction](#) ()

*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for an iconst instruction with no parameters.*

- boolean [correct](#) ()

*Simple instruction line is correct if it has no parameter.*

### Static Public Member Functions

- static String[] [getMnemonics](#) ()

*This method returns the array of iconst [instructions](#) mnemonics.*

### Private Member Functions

- Instruction [getIConstInstruction](#) (finalInstruction a\_res)

*This method creates the objects that represent iconst [instructions](#) (e.g.*

## Static Private Attributes

- static final int **MAX\_ICONST\_NUM** = 5

*The constant that represents the maximal value of the constant parameter for [instructions](#) such as `iconst_<n>`, see [getConstInstruction\(Instruction\)](#) for the full inventory.*

### 6.55.1 Detailed Description

This class handles the creation and correctness for `iconst` [instructions](#) with no parameters.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.55.2 Constructor & Destructor Documentation

#### 6.55.2.1 **umbra.instructions.ast.IConstInstruction.IConstInstruction (final String *a\_line\_text*, final String *a\_name*)**

This creates an instance of an instruction named as *a\_name* with the line text *a\_line*.

Currently it just calls the constructor of the superclass.

#### Parameters:

*a\_line\_text* the line number of the instruction  
*a\_name* the mnemonic name of the instruction

#### See also:

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.55.3 Member Function Documentation

#### 6.55.3.1 **static String [ ] umbra.instructions.ast.IConstInstruction.getMnemonics () [static]**

This method returns the array of `iconst` [instructions](#) mnemonics.

#### Returns:

the array of the handled mnemonics

#### See also:

[InstructionLineController](#).[getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

### 6.55.3.2 Instruction `umbra.instructions.ast.IConstInstruction.getIConstInstruction (final Instruction a_res) [private]`

This method creates the objects that represent `iconst` [instructions](#) (e.g. `iconst_0`). It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

#### Parameters:

`a_res` a helper value returned in case the current instruction is not in the current set

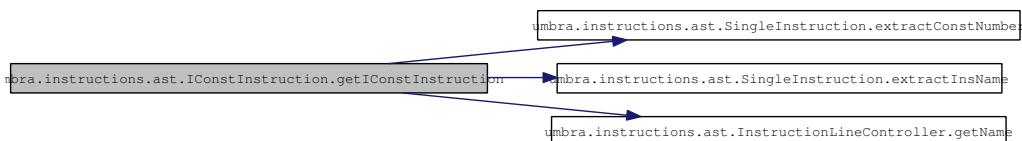
#### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.SingleInstruction.extractConstNumber()`, `umbra.instructions.ast.SingleInstruction.extractInsName()`, `umbra.instructions.ast.InstructionLineController.getName()`, and `umbra.instructions.ast.IConstInstruction.MAX_ICONST_NUM`.

Referenced by `umbra.instructions.ast.IConstInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.55.3.3 Instruction `umbra.instructions.ast.IConstInstruction.getInstruction ()`

This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for an `iconst` instruction with no parameters.

The method can construct an instruction from `iconst` [instructions](#) only.

This method also checks the syntactical correctness of the current instruction line.

#### Returns:

the freshly constructed BCEL instruction or `null` in case the instruction is not an `iconst` instruction and in case the instruction line is incorrect

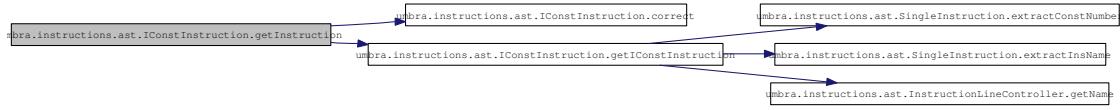
#### See also:

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from `umbra.instructions.ast.SingleInstruction`.

References `umbra.instructions.ast.IConstInstruction.correct()`, and `umbra.instructions.ast.IConstInstruction.getIConstInstruction()`.

Here is the call graph for this function:



#### 6.55.3.4 boolean umbra.instructions.ast.IConstInstruction.correct ()

Simple instruction line is correct if it has no parameter.

**Returns:**

true when the instruction mnemonic is the only text in the line of the instruction text

**See also:**

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

Referenced by [umbra.instructions.ast.IConstInstruction.getInstruction\(\)](#).

#### 6.55.4 Member Data Documentation

##### 6.55.4.1 final int umbra.instructions.ast.IConstInstruction.MAX\_ICONST\_NUM = 5 [static, private]

The constant that represents the maximal value of the constant parameter for [instructions](#) such as `iconst_-<n>`, see [getIConstInstruction\(Instruction\)](#) for the full inventory.

Referenced by [umbra.instructions.ast.IConstInstruction.getIConstInstruction\(\)](#).

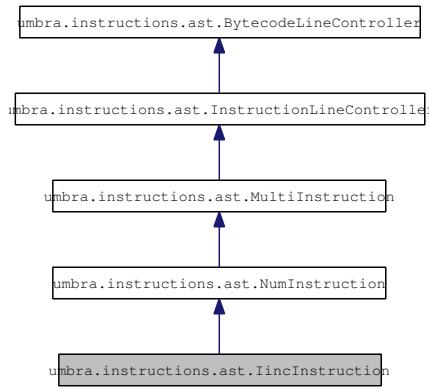
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[IConstInstruction.java](#)

## 6.56 umbra.instructions.ast.IincInstruction Class Reference

This class handles the creation and correctness for iinc instruction.

Inheritance diagram for umbra.instructions.ast.IincInstruction:



Collaboration diagram for umbra.instructions.ast.IincInstruction:



### Public Member Functions

- **IincInstruction** (final String a\_line\_text, final String a\_name)  
*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line.*
- final boolean **correct** ()  
*Inc instruction line is correct if it has two simple number parameters (first preceded with %).*
- final Instruction **getInstruction** ()  
*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for the iinc instruction.*

### Static Public Member Functions

- static String[] **getMnemonics** ()  
*This method returns the array of inc **instructions** mnemonics.*

### Private Member Functions

- int **getInd1** ()  
*This method parses the first parameter of the current instruction.*

- int `getInd2()`

*This method parses the second parameter of the current instruction.*

### 6.56.1 Detailed Description

This class handles the creation and correctness for iinc instruction.

**Author:**

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.56.2 Constructor & Destructor Documentation

#### 6.56.2.1 `umbra.instructions.ast.IincInstruction.IincInstruction (final String a_line_text, final String a_name)`

This creates an instance of an instruction named as `a_name` in a line the text of which is `a_line`.

Currently it just calls the constructor of the superclass.

**Parameters:**

`a_line_text` the line number of the instruction  
`a_name` the mnemonic name of the instruction

**See also:**

`InstructionLineControllerInstructionLineController(String, String)`

### 6.56.3 Member Function Documentation

#### 6.56.3.1 `static String [ ] umbra.instructions.ast.IincInstruction.getMnemonics () [static]`

This method returns the array of inc `instructions` mnemonics.

**Returns:**

the array of the handled mnemonics

**See also:**

`InstructionLineController.getMnemonics()`

Reimplemented from `umbra.instructions.ast.InstructionLineController`.

### 6.56.3.2 final boolean umbra.instructions.ast.IincInstruction.correct ()

Inc instruction line is correct if it has two simple number parameters (first preceded with %).

The precise format is as follows: whitespace number : whitespace mnemonic whitespace % number whitespace number whitespace lineend

#### Returns:

true when the syntax of the instruction line is correct

#### See also:

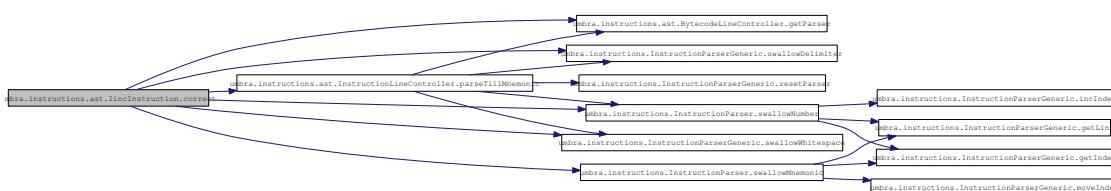
[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

References                [umbra.instructions.ast.BytecodeLineController.getParser\(\)](#),                um-  
[umbra.instructions.ast.InstructionLineController.parseTillMnemonic\(\)](#),                um-  
[umbra.instructions.InstructionParserGeneric.swallowDelimiter\(\)](#), [umbra.instructions.InstructionParser.swallowMnemonic\(\)](#),  
[umbra.instructions.InstructionParser.swallowNumber\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#)

Referenced by [umbra.instructions.ast.IincInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



### 6.56.3.3 int umbra.instructions.ast.IincInstruction.getInd1 () [private]

This method parses the first parameter of the current instruction.

This method retrieves the numerical value of the parameter of the instruction in [BytecodeLineController#getMy\\_line\\_text\(\)](#). This parameter is located after the mnemonic followed by % (with no whitespace inbetween). The method assumes [BytecodeLineController#getMy\\_line\\_text\(\)](#) is correct.

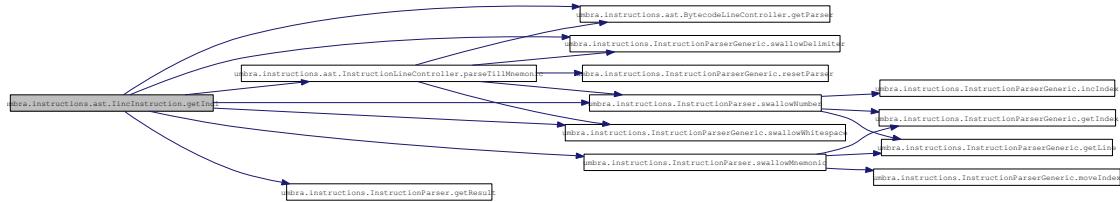
#### Returns:

the value of the numerical parameter of the instruction

References                [umbra.instructions.ast.BytecodeLineController.getParser\(\)](#),                um-  
[umbra.instructions.InstructionParser.getResult\(\)](#), [umbra.instructions.ast.InstructionLineController.parseTillMnemonic\(\)](#),  
[umbra.instructions.InstructionParserGeneric.swallowDelimiter\(\)](#), [umbra.instructions.InstructionParser.swallowMnemonic\(\)](#),  
[umbra.instructions.InstructionParser.swallowNumber\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#)

Referenced by [umbra.instructions.ast.IincInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



#### 6.56.3.4 int umbra.instructions.ast.IincInstruction.getInd2 () [private]

This method parses the second parameter of the current instruction.

This method retrieves the numerical value of the parameter of the instruction in [BytecodeLineController#getMy\\_line\\_text\(\)](#). This parameter is located after the first parameter (with some whitespace inbetween). The method assumes [BytecodeLineController#getMy\\_line\\_text\(\)](#) is correct. It also assumes that the internal parser state has not been modified between the call to [getInd1\(\)](#) and the call of this method.

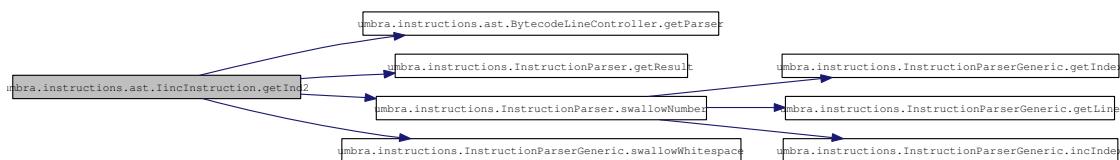
**Returns:**

the value of the second numerical parameter of the instruction

References [umbra.instructions.ast.BytecodeLineController.getParser\(\)](#), [umbra.instructions.InstructionParser.getResult\(\)](#), [umbra.instructions.InstructionParser.swallowNumber\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#).

Referenced by [umbra.instructions.ast.IincInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



#### 6.56.3.5 final Instruction umbra.instructions.ast.IincInstruction.getInstruction ()

This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for the iinc instruction.

It computes the parameters of the instruction before the instruction is constructed. This method also checks the syntactical correctness of the current instruction line.

**Returns:**

the freshly constructed BCEL instruction or `null` in case the instruction is not the iinc instruction and in case the instruction line is incorrect

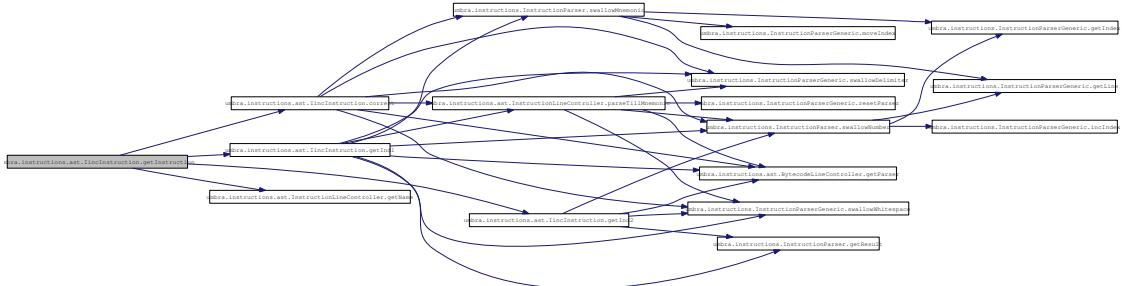
**See also:**

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References `umbra.instructions.ast.IincInstruction.correct()`, `umbra.instructions.ast.IincInstruction.getInd1()`, `umbra.instructions.ast.IincInstruction.getInd2()`, and `umbra.instructions.ast.InstructionLineController.getName()`.

Here is the call graph for this function:



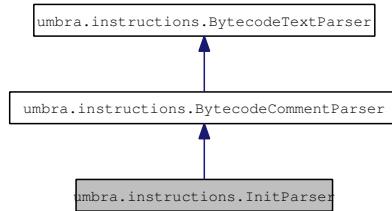
The documentation for this class was generated from the following file:

- [source/umbra/instructions/ast/IincInstruction.java](#)

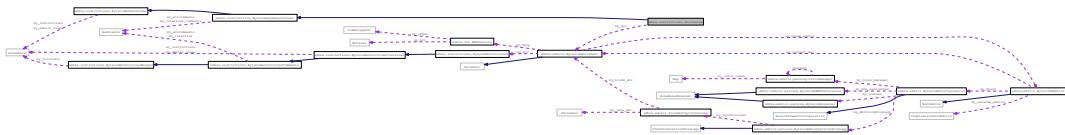
## 6.57 umbra.instructions.InitParser Class Reference

This class handles the initial parsing of a byte code textual document.

Inheritance diagram for `umbra.instructions.InitParser`:



Collaboration diagram for `umbra.instructions.InitParser`:



### Public Member Functions

- `InitParser (final BytecodeDocument a_doc, final String[ ] a_comment_array, final String[ ] a_interline)`  
*This constructor initialises all the internal structures.*
- `final String runParsing () throws UmbraLocationException, UmbraMethodException`  
*Initialisation of all the byte code structures related to the document; it uses BCEL objects associated with the document and based on them it generates the Umbra line structures (subclasses of the `BytecodeLineController`) together with the links to the BCEL objects.*

### Private Member Functions

- `int swallowClassHeader (final int the_current_line, final LineContext a_ctxt) throws UmbraLocationException`  
*The method parses the initial portion of a byte code text.*
- `int swallowMethod (final int the_line_no, final int a_method_no, final LineContext a_ctxt) throws UmbraLocationException, UmbraMethodException`  
*This method handles the parsing of these lines of a textual representation which contain a method.*
- `int swallowMethodHeader (final LineContext a_ctxt, final int a_lineno, final MethodGen a_methodgen) throws UmbraLocationException`  
*This method handles the parsing of the method header lines.*

## Private Attributes

- `BytecodeDocument my_doc`

*The byte code document to be parsed.*

### 6.57.1 Detailed Description

This class handles the initial parsing of a byte code textual document.

It creates handlers for each line of the document and structures to handle the end-of-line comments. It is also able to reconstruct the end-of-line comments from the previous session (closed with the refresh action).

This class is used by `BytecodeController` to initialise its internal structures at the beginning of editing or after the refresh action is performed.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Tomek Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

### 6.57.2 Constructor & Destructor Documentation

#### 6.57.2.1 `umbra.instructions.InitParser` (final `BytecodeDocument a_doc`, final `String[ ] a_comment_array`, final `String[ ] a_interline`)

This constructor initialises all the internal structures.

It memorises the given document and array with end-of-line comments. Furthermore, it sets all the internal containers to be empty.

#### Parameters:

`a_doc` the byte code document with the corresponding BCEL structures linked into it

`a_comment_array` contains the texts of end-of-line comments, the i-th entry contains the comment for the i-th instruction in the document, if this parameter is null then the array is not taken into account

`a_interline` contains the texts of interline comments, the i-th entry contains the comment for the i-th instruction in the document, if this parameter is null then the array is not taken into account

References `umbra.instructions.InitParser.my_doc`.

### 6.57.3 Member Function Documentation

#### 6.57.3.1 final String umbra.instructions.InitParser.runParsing () throws UmbraLocationException, UmbraMethodException

Initialisation of all the byte code structures related to the document; it uses BCEL objects associated with the document and based on them it generates the Umbra line structures (subclasses of the [BytecodeLineController](#)) together with the links to the BCEL objects.

The comment structures that might have come from previous sessions may cause changes in the original textual representation. The method returns the changed representation.

This method initialises the parsing context, then it parses the header of the class and then one by one parses the methods. At the end the method initialises the structures to keep track of the modified methods.

##### Returns:

changed textual representation of the parsed class

##### Exceptions:

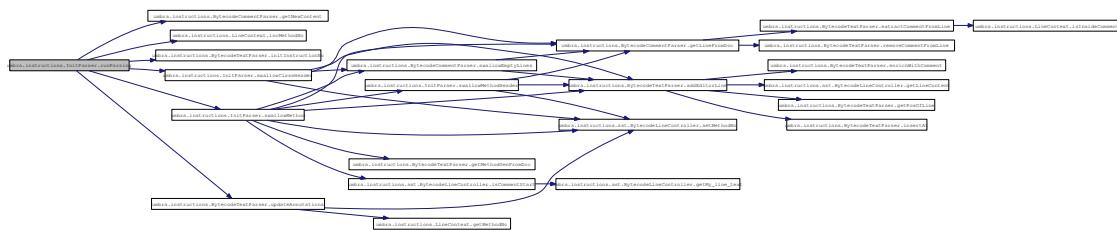
**UmbraLocationException** thrown in case a position has been reached which is outside the current document

**UmbraMethodException** thrown in case a method number has been reached which is outside the number of available methods in the document

References      `umbra.instructions.BytecodeCommentParser.getNewContent()`, `umbra.instructions.LineContext.incMethodNo()`, `umbra.instructions.BytecodeTextParser.initInstructionNo()`, `umbra.instructions.InitParser.my_doc`, `umbra.instructions.InitParser.swallowClassHeader()`, `umbra.instructions.InitParser.swallowMethod()`, and `umbra.instructions.BytecodeTextParser.updateAnnotations()`.

Referenced by `umbra.instructions.BytecodeControllerContainer.init()`.

Here is the call graph for this function:



#### 6.57.3.2 int umbra.instructions.InitParser.swallowClassHeader (final int *the\_current\_line*, final LineContext *a\_ctxt*) throws UmbraLocationException [private]

The method parses the initial portion of a byte code text.

This portion contains the information about the class which the code implements. The exact format is:

```
public PackageName
[ emptylines ]
AccessModifier class ClassName
[ emptylines ]
```

Note that emptylines may be comments as well.

**Parameters:**

- the\_current\_line* the line from which we start the parsing (mostly 0)
- a\_ctxt* the parsing context

**Returns:**

the advanced line number, the first line number which has not been analysed by the current method

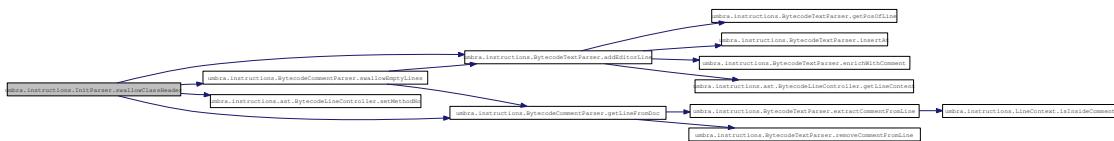
**Exceptions:**

**UmbraLocationException** in case one of the locations in the document was wrongly calculated

References                   umbra.instructions.BytecodeTextParser.addEditorLine(),                   um-  
 bra.instructions.BytecodeCommentParser.getLineFromDoc(),                   umbra.instructions.InitParser.my\_-  
 doc,                   umbra.instructions.ast.BytecodeLineController.setMethodNo(),                   and           um-  
 bra.instructions.BytecodeCommentParser.swallowEmptyLines().

Referenced by `umbra.instructions.InitParser.runParsing()`.

Here is the call graph for this function:



### 6.57.3.3 int umbra.instructions.InitParser.swallowMethod (final int *the\_line\_no*, final int *a\_method\_no*, final LineContext *a\_ctxt*) throws UmbraLocationException, UmbraMethodException [private]

This method handles the parsing of these lines of a textual representation which contain a method.

The method first swallows the eventual empty lines before the method. Then the method checks if the method currently to be parsed can fit into the structures within the BCEL representation. Subsequently it parses line by line the given document starting with the given line and tries to parse the lines and associate with them the [instructions](#) from the BCEL structures. It assumes that the method starts with a method header. The current method ends when an empty line is met or when the end of the document is reached.

**Parameters:**

- the\_line\_no* the line in the document starting with which the method parsing begins
- a\_method\_no* the number of the method to be parsed
- a\_ctxt* a parsing context

**Returns:**

the number of the first line after the method; it is the first line after the empty method delimiting line or the last line in the document in case the end of document is met

**Exceptions:**

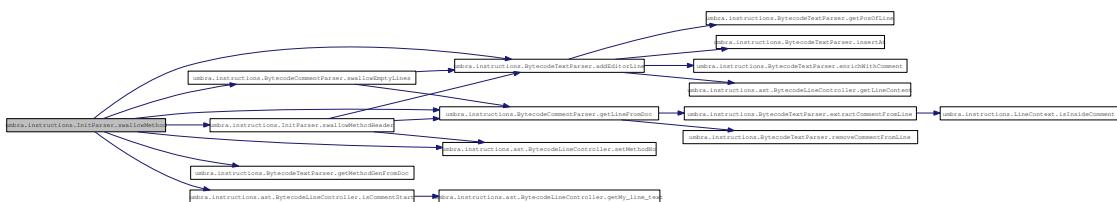
**UmbraLocationException** in case a line number is reached which is not within the given document

**UmbraMethodException** the given method number exceeds the range of available methods in the BCEL structure

References      `umbra.instructions.BytecodeTextParser.addEditorLine()`, `umbra.instructions.BytecodeCommentParser.getLineFromDoc()`, `umbra.instructions.BytecodeTextParser.getMethodGenFromDoc()`, `umbra.instructions.ast.BytecodeLineController.isCommentStart()`, `umbra.instructions.InitParser.my_doc`, `umbra.instructions.ast.BytecodeLineController.setMethodNo()`, `umbra.instructions.BytecodeCommentParser.swallowEmptyLines()`, `and` `umbra.instructions.InitParser.swallowMethodHeader()`.

Referenced by `umbra.instructions.InitParser.runParsing()`.

Here is the call graph for this function:



#### 6.57.3.4 int umbra.instructions.InitParser.swallowMethodHeader (final LineContext *a\_ctxt*, final int *a\_lineno*, final MethodGen *a\_methodgen*) throws UmbraLocationException [private]

This method handles the parsing of the method header lines.

It assumes that the header contains the method signature and possibly the throws declarations. The method finishes its work on the first non-throws line of the document.

##### Parameters:

- a\_ctxt* the parsing context with which the parsing is done
- a\_lineno* the line number of the first line to be parsed
- a\_methodgen* the BCEL method representation

##### Returns:

the number of the first line that could not be parsed by this method

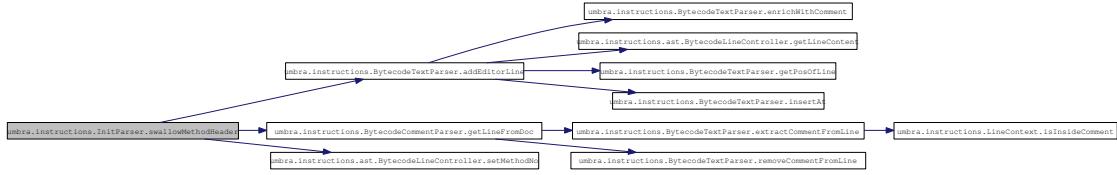
##### Exceptions:

**UmbraLocationException** in case a line number has been reached such that there is no such a line in the current document

References      `umbra.instructions.BytecodeTextParser.addEditorLine()`, `umbra.instructions.BytecodeCommentParser.getLineFromDoc()`, `umbra.instructions.InitParser.my_doc`, and `umbra.instructions.ast.BytecodeLineController.setMethodNo()`.

Referenced by `umbra.instructions.InitParser.swallowMethod()`.

Here is the call graph for this function:



## 6.57.4 Member Data Documentation

### 6.57.4.1 BytecodeDocument umbra.instructions.InitParser.my\_doc [private]

The byte code document to be parsed.

It contains the corresponding BCEL structures linked into it.

Referenced by `umbra.instructions.InitParser()`, `umbra.instructions.InitParser.runParsing()`, `umbra.instructions.InitParser.swallowClassHeader()`, `umbra.instructions.InitParser.swallowMethod()`, and `umbra.instructions.InitParser.swallowMethodHeader()`.

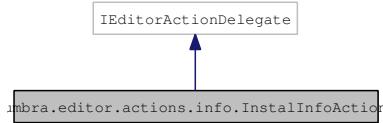
The documentation for this class was generated from the following file:

- source/umbra/instructions/[InitParser.java](#)

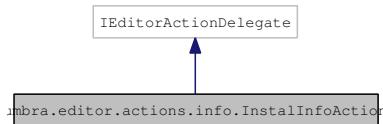
## 6.58 umbra.editor.actions.info.InstalInfoAction Class Reference

The class implements the behaviour in case the Install Info button in the bytecode [editor](#) is pressed.

Inheritance diagram for `umbra.editor.actions.info.InstalInfoAction`:



Collaboration diagram for `umbra.editor.actions.info.InstalInfoAction`:



### Public Member Functions

- final void  [setActiveEditor](#) (final `IAction` `an_action`, final `IEditionPart` `a_target_editor`)  
*The method sets the `editor` associated with the action.*
- final void  [run](#) (final `IAction` `an_action`)  
*The method shows the content of the install `info instructions`.*
- void  [selectionChanged](#) (final `IAction` `an_action`, final `ISelection` `a_selection`)  
*The method reacts when the selected area changes in the bytecode `editor`.*

### 6.58.1 Detailed Description

The class implements the behaviour in case the Install Info button in the bytecode [editor](#) is pressed.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

#### Version:

a-01

### 6.58.2 Member Function Documentation

#### 6.58.2.1 final void `umbra.editor.actions.info.InstalInfoAction.setActiveEditor` (final `IAction` `an_action`, final `IEditionPart` `a_target_editor`)

The method sets the `editor` associated with the action.

**Parameters:**

*an\_action* ignored  
*a\_target\_editor* ignored

**6.58.2.2 final void umbra.editor.actions.info.InstalInfoAction.run (final IAction *an\_action*)**

The method shows the content of the install [info instructions](#).

Currently, it only pops up the general help browser.

FIXME the method should open something more specific, note that it is tricky to know the proper ID to open it should open something like Info/info.txt <https://mobius.ucd.ie/ticket/557>

**Parameters:**

*an\_action* action that triggered the showing of the instruction

**6.58.2.3 void umbra.editor.actions.info.InstalInfoAction.selectionChanged (final IAction *an\_action*, final ISelection *a\_selection*)**

The method reacts when the selected area changes in the bytecode [editor](#).

Currently, it does nothing.

**Parameters:**

*an\_action* the action which triggered the selection change  
*a\_selection* the new selection.

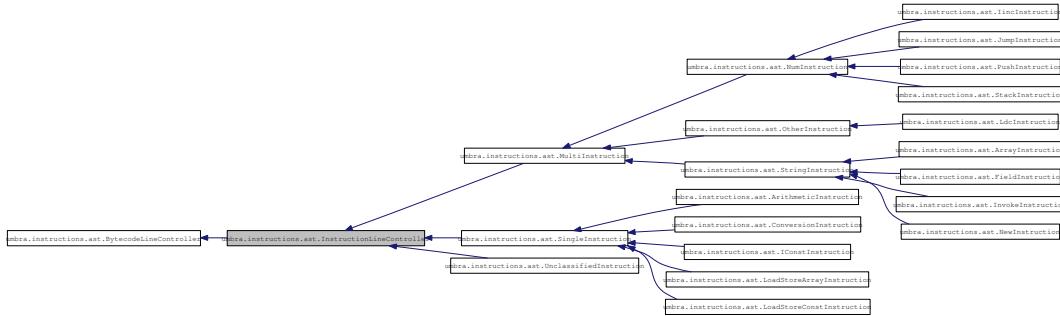
The documentation for this class was generated from the following file:

- source/umbra/editor/actions/info/[InstalInfoAction.java](#)

## 6.59 umbra.instructions.ast.InstructionLineController Class Reference

This class defines a structure that describes a single byte code instruction and contains related BCEL structures.

Inheritance diagram for `umbra.instructions.ast.InstructionLineController`:



Collaboration diagram for `umbra.instructions.ast.InstructionLineController`:



## Public Member Functions

- **`InstructionLineController`** (final String a\_line\_text, final String a\_name)
 

*The construction creates the controller which binds the instruction mnemonic with the line text.*
- final boolean **`addHandle`** (final `InstructionHandle` a\_handle, final `InstructionList` a\_list, final `MethodGen` a\_method\_gen)
 

*The method adds the link between the Umbra representation of `instructions` to their representation in BCEL.*
- final `InstructionHandle` **`getHandle`** ()
 

*Returns the `InstructionHandle` structure which corresponds to the current instruction.*
- final `InstructionList` **`getList`** ()
 

*Returns the `InstructionList` structure in which the current instruction is located.*
- final `MethodGen` **`getMethod`** ()
 

*Returns the `MethodGen` structure responsible for the method in which the instruction resides.*
- boolean **`correct`** ()
 

*This method is redefined in each subclass.*
- String **`getName`** ()
 

*Returns the name of the mnemonic.*

- boolean `replace` (final `InstructionLineController` a\_newlc)

*This method replaces the current instruction handle in the method generation structure with the one for the given instruction.*

## Static Public Member Functions

- static String[ ] `getMnemonics` ()

*This method returns the array of mnemonics handled by the current class.*

- static void `controlPrint` (final `BytecodeLineController` a\_line)

*The debugging method that prints out to the standard output the information on the line given in the parameter.*

- static void `printInstructionList` (final `InstructionList` an\_ilist)

*This is a debugging helper method which prints out to the standard output the contents of the given BCEL instruction list.*

## Static Public Attributes

- static final Class[ ] `INS_CLASS_HIERARCHY`

*This array contains the classes which are able to handle lines with mnemonics.*

## Protected Member Functions

- void `setName` (final String a\_name)
- boolean `parseTillMnemonic` ()

*This method parses initial part of a instruction line.*

## Static Private Member Functions

- static void `addTargeters` (final `InstructionHandle` an\_nins, final `InstructionHandle` an\_oins, final `InstructionTargeter`[ ] the\_trgtrs)

*This method adds given `InstructionTargeter` objects to the given instruction.*

## Private Attributes

- `InstructionList` `my_instr_list`

*The list of `instructions` in the method to which the current instruction belongs.*

- `InstructionHandle` `my_instr_handle`

*A BCEL handle to the current instruction representation in BCEL format.*

- `MethodGen` `my_methodgen`

A BCEL object that represents the method in which the current instruction is located.

- String `my_name`  
*The mnemonic name of the current instruction.*

### 6.59.1 Detailed Description

This class defines a structure that describes a single byte code instruction and contains related BCEL structures.

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Tomek Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.59.2 Constructor & Destructor Documentation

#### 6.59.2.1 `umbra.instructions.ast.InstructionLineController.InstructionLineController (final String a_line_text, final String a_name)`

The construction creates the controller which binds the instruction mnemonic with the line text.

The name is set locally while the assignment of the line is done in the constructor of the superclass.

**Parameters:**

`a_line_text` the string representation of the line text  
`a_name` the mnemonic name of the instruction

**See also:**

`BytecodeLineController`.`BytecodeLineController(String)`

References `umbra.instructions.ast.InstructionLineController.setName()`.

Here is the call graph for this function:



### 6.59.3 Member Function Documentation

#### 6.59.3.1 `static String [] umbra.instructions.ast.InstructionLineController.getMnemonics () [static]`

This method returns the array of mnemonics handled by the current class.

**Returns:**

the array of the handled mnemonics

Reimplemented in [umbra.instructions.ast.ArithmeticInstruction](#), [umbra.instructions.ast.ArrayInstruction](#), [umbra.instructions.ast.ConversionInstruction](#), [umbra.instructions.ast.FieldInstruction](#), [umbra.instructions.ast.IConstInstruction](#), [umbra.instructions.ast.IincInstruction](#), [umbra.instructions.ast.InvokeInstruction](#), [umbra.instructions.ast.JumpInstruction](#), [umbra.instructions.ast.LdcInstruction](#), [umbra.instructions.ast.LoadStoreArrayInstruction](#), [umbra.instructions.ast.LoadStoreConstInstruction](#), [umbra.instructions.ast.NewInstruction](#), [umbra.instructions.ast.PushInstruction](#), [umbra.instructions.ast.SingleInstruction](#), [umbra.instructions.ast.StackInstruction](#), and [umbra.instructions.ast.UnclassifiedInstruction](#).

#### **6.59.3.2 final boolean umbra.instructions.ast.InstructionLineController.addHandle (final InstructionHandle a\_handle, final InstructionList a\_list, final MethodGen a\_method\_gen)**

The method adds the link between the Umbra representation of [instructions](#) to their representation in BCEL.

**Parameters:**

*a\_handle* the BCEL instruction handle that corresponds to the instruction associated with the current object

*a\_list* the list of [instructions](#) in the current method

*a\_method\_gen* the object which represents the method of the current instruction in the BCEL representation of the current class in the byte code [editor](#)

**Returns:**

always true as the subclasses of the current class correspond to [instructions](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

Referenced by [umbra.instructions.ast.InstructionLineController.replace\(\)](#).

#### **6.59.3.3 static void umbra.instructions.ast.InstructionLineController.controlPrint (final BytecodeLineController a\_line) [static]**

The debugging method that prints out to the standard output the information on the line given in the parameter.

It prints out:

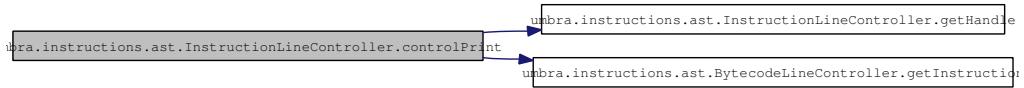
- the name of the instruction,
- the position of the instruction handle

**Parameters:**

*a\_line* the line for which the information is printed out

References [umbra.instructions.ast.InstructionLineController.getHandle\(\)](#), and [umbra.instructions.ast.BytecodeLineController.getInstruction\(\)](#).

Here is the call graph for this function:



#### **6.59.3.4 static void umbra.instructions.ast.InstructionLineController.printInstructionList (final InstructionList an\_ilist) [static]**

This is a debugging helper method which prints out to the standard output the contents of the given BCEL instruction list.

**Parameters:**

*an\_ilist* the instruction list to print out

#### **6.59.3.5 final InstructionHandle umbra.instructions.ast.InstructionLineController.getHandle ()**

Returns the [InstructionHandle](#) structure which corresponds to the current instruction.

**Returns:**

the BCEL handle to the current instruction.

References [umbra.instructions.ast.InstructionLineController.my\\_instr\\_handle](#).

Referenced by [umbra.instructions.BytecodeControllerContainer.controlPrint\(\)](#), and [umbra.instructions.ast.InstructionLineController.controlPrint\(\)](#).

#### **6.59.3.6 final InstructionList umbra.instructions.ast.InstructionLineController.getList ()**

Returns the [InstructionList](#) structure in which the current instruction is located.

**Returns:**

the BCEL list of the [instructions](#) of the method to which the current instruction belongs

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References [umbra.instructions.ast.InstructionLineController.my\\_instr\\_list](#).

#### **6.59.3.7 final MethodGen umbra.instructions.ast.InstructionLineController.getMethod ()**

Returns the [MethodGen](#) structure responsible for the method in which the instruction resides.

**Returns:**

the method in which the current instruction is located

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References [umbra.instructions.ast.InstructionLineController.my\\_methodgen](#).

Referenced by [umbra.instructions.ast.InstructionLineController.replace\(\)](#).

### 6.59.3.8 boolean umbra.instructions.ast.InstructionLineController.correct ()

This method is redefined in each subclass.

It is used to check some basic condition of correctness. A positive result is necessary to continue any attempt of generating BCEL structures about the line.

#### Returns:

true if the instruction is correct

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

Reimplemented in [umbra.instructions.ast.ArithmeticInstruction](#), [umbra.instructions.ast.ArrayInstruction](#), [umbra.instructions.ast.ConversionInstruction](#), [umbra.instructions.ast.FieldInstruction](#), [umbra.instructions.ast.IConstInstruction](#), [umbra.instructions.ast.IincInstruction](#), [umbra.instructions.ast.InvokeInstruction](#), [umbra.instructions.ast.JumpInstruction](#), [umbra.instructions.ast.LdcInstruction](#), [umbra.instructions.ast.LoadStoreArrayInstruction](#), [umbra.instructions.ast.LoadStoreConstInstruction](#), [umbra.instructions.ast.NewInstruction](#), [umbra.instructions.ast.PushInstruction](#), [umbra.instructions.ast.SingleInstruction](#), [umbra.instructions.ast.StackInstruction](#), and [umbra.instructions.ast.UnclassifiedInstruction](#).

### 6.59.3.9 void umbra.instructions.ast.InstructionLineController.setName (final String *a\_name*) [protected]

#### Parameters:

*a\_name* the mnemonic name to set

References [umbra.instructions.ast.InstructionLineController.my\\_name](#).

Referenced by [umbra.instructions.ast.InstructionLineController.InstructionLineController\(\)](#).

### 6.59.3.10 String umbra.instructions.ast.InstructionLineController.getName ()

Returns the name of the mnemonic.

#### Returns:

the name of the mnemonic

References [umbra.instructions.ast.InstructionLineController.my\\_name](#).

Referenced by [umbra.instructions.BytecodeControllerContainer.controlPrint\(\)](#), [umbra.instructions.ast.StackInstruction.getAInstruction\(\)](#), [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayArrayLSInstruction\(\)](#), [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayBoolLSInstruction\(\)](#), [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayCharLSInstruction\(\)](#), [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayDoubleLSInstruction\(\)](#), [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayFloatLSInstruction\(\)](#), [umbra.instructions.ast.SingleInstruction.getArrayInstruction\(\)](#), [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayIntLSInstruction\(\)](#), [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLongLSInstruction\(\)](#), [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayShortLAInstruction\(\)](#), [umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction\(\)](#), [umbra.instructions.ast.ConversionInstruction.getD2XConvOp\(\)](#), [umbra.instructions.ast.StackInstruction.getDInstruction\(\)](#), [umbra.instructions.ast.ArithmeticInstruction.getDOpInstruction\(\)](#), [umbra.instructions.ast.SingleInstruction.getDupInstruction\(\)](#).

umbra.instructions.ast.ConversionInstruction.getF2XConvOp(), umbra.instructions.ast.StackInstruction.getFIInstruction(),  
 umbra.instructions.ast.ArithmeticInstruction.getFOpInstruction(), umbra.instructions.ast.JumpInstruction.getGotoInstruction(),  
 umbra.instructions.ast.ConversionInstruction.getI2XConvOp(), umbra.instructions.ast.ArithmeticInstruction.getIBoolOpInstruction(),  
 umbra.instructions.ast.IConstInstruction.getIConstInstruction(), umbra.instructions.ast.StackInstruction.getIIInstruction(),  
 umbra.instructions.ast.SingleInstruction.getInstruction(), umbra.instructions.ast.PushInstruction.getInstruction(),  
 umbra.instructions.ast.NewInstruction.getInstruction(), umbra.instructions.ast.LdcInstruction.getInstruction(),  
 umbra.instructions.ast.InvokeInstruction.getInstruction(), umbra.instructions.ast.IincInstruction.getInstruction(),  
 umbra.instructions.ast.FieldInstruction.getInstruction(), umbra.instructions.ast.ArrayInstruction.getInstruction(),  
 umbra.instructions.ast.JumpInstruction.getIntCompIfInstruction(), umbra.instructions.ast.ArithmeticInstruction.getIOpInstruction(),  
 umbra.instructions.ast.ConversionInstruction.getL2XConvOp(), umbra.instructions.ast.ArithmeticInstruction.getLBoolOpInstruction(),  
 umbra.instructions.ast.StackInstruction.getLIInstruction(), umbra.instructions.ast.ArithmeticInstruction.getLOpInstruction(),  
 umbra.instructions.ast.ArithmeticInstruction.getLShiftOpInstruction(),  
 bra.instructions.ast.SingleInstruction.getMonitorInstruction(), umbra.instructions.ast.JumpInstruction.getNullCompIfInstruction(),  
 umbra.instructions.ast.JumpInstruction.getRefCompIfInstruction(),  
 bra.instructions.ast.SingleInstruction.getReturnInstruction(), umbra.instructions.ast.JumpInstruction.getSubroutineInstruction(),  
 umbra.instructions.ast.SingleInstruction.getTopManipulationInstruction(), and  
 bra.instructions.ast.JumpInstruction.getZeroCompIfInstruction().

#### 6.59.3.11 boolean umbra.instructions.ast.InstructionLineController.parseTillMnemonic () [protected]

This method parses initial part of a instruction line.

This is a helper method which parses the common part of each instruction line i.e.:

whitespace number : whitespace

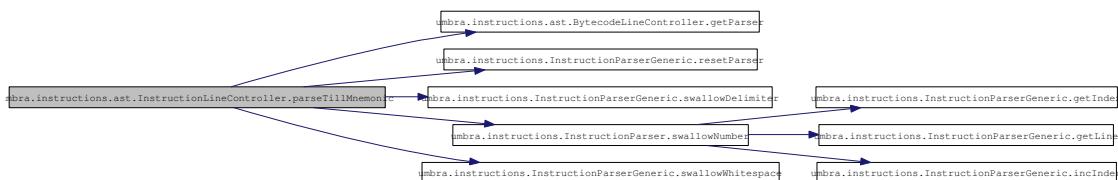
#### Returns:

true when all the parsing is done sucessfully, false in case the initial portion of the line is not of the required form

References      umbra.instructions.ast.BytecodeLineController.getParser(),  
 umbra.instructions.InstructionParserGeneric.resetParser(), umbra.instructions.InstructionParserGeneric.swallowDelimiter(),  
 umbra.instructions.InstructionParser.swallowNumber(), and umbra.instructions.InstructionParserGeneric.swallowWhitespace()

Referenced by      umbra.instructions.ast.NumInstruction.checkInstructionWithNumber(),  
 umbra.instructions.ast.StackInstruction.correct(), umbra.instructions.ast.PushInstruction.correct(),  
 umbra.instructions.ast.NewInstruction.correct(), umbra.instructions.ast.LdcInstruction.correct(),  
 umbra.instructions.ast.JumpInstruction.correct(), umbra.instructions.ast.InvokeInstruction.correct(),  
 umbra.instructions.ast.IincInstruction.correct(), umbra.instructions.ast.FieldInstruction.correct(),  
 umbra.instructions.ast.ArrayInstruction.correct(), and umbra.instructions.ast.IincInstruction.getInd1().

Here is the call graph for this function:



### 6.59.3.12 boolean umbra.instructions.ast.InstructionLineController.replace (final InstructionLineController *a\_newlc*)

This method replaces the current instruction handle in the method generation structure with the one for the given instruction.

First, we check if the given new line controller can give a proper BCEL representation of an instruction. If it cannot, `false` is returned. Next we check if the current instruction is the first one in the method. Depending on that we insert the new instruction either at the beginning of the method or after the instruction right before the current one (respectively). In case the current instruction is a target of some other `instructions` in the method, we re-target them to the new instruction. At last, we delete the current instruction from the instruction list of the current method.

The current instruction line controller should not be used after the call to this method as it is disconnected from the BCEL structures.

#### Parameters:

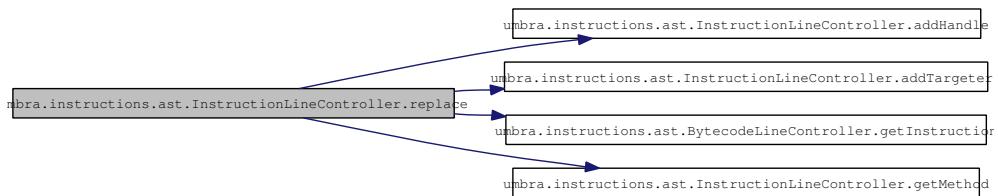
*a\_newlc* the instruction line which should replace the current one

#### Returns:

`true` if the operation was carried out successfully, `false` otherwise

References      `umbra.instructions.ast.InstructionLineController.addHandle()`,      `umbra.instructions.ast.InstructionLineController.addTargeters()`, `umbra.instructions.ast.BytecodeLineController.getInstruction()`, `umbra.instructions.ast.InstructionLineController.getMethod()`, and `umbra.instructions.ast.InstructionLineController.my_instr_handle`.

Here is the call graph for this function:



### 6.59.3.13 static void umbra.instructions.ast.InstructionLineController.addTargeters (final InstructionHandle *an\_nins*, final InstructionHandle *an\_oins*, final InstructionTargeter[] *the\_trgtrs*) [static, private]

This method adds given `InstructionTargeter` objects to the given instruction.

#### Parameters:

*an\_nins* the `InstructionHandle` to add the targeters to

*an\_oins* the `InstructionHandle` to be replaced in targeters with the new one

*the\_trgtrs* the array with targeters to add to the instruction

Referenced by `umbra.instructions.ast.InstructionLineController.replace()`.

## 6.59.4 Member Data Documentation

### 6.59.4.1 final Class [ ] **umbra.instructions.ast.InstructionLineController.INS\_CLASS\_HIERARCHY** [static]

**Initial value:**

```
{
    ArithmeticInstruction.class,
    IConstInstruction.class,
    LoadStoreConstInstruction.class,
    LoadStoreArrayInstruction.class,
    ConversionInstruction.class,
    SingleInstruction.class,
    PushInstruction.class,
    JumpInstruction.class,
    IincInstruction.class,
    StackInstruction.class,
    ArrayInstruction.class,
    NewInstruction.class,
    FieldInstruction.class,
    InvokeInstruction.class,
    LdcInstruction.class,
    UnclassifiedInstruction.class}
```

This array contains the classes which are able to handle lines with mnemonics.

### 6.59.4.2 InstructionList **umbra.instructions.ast.InstructionLineController.my\_instr\_list** [private]

The list of [instructions](#) in the method to which the current instruction belongs.

Referenced by `umbra.instructions.ast.InstructionLineController.getList()`.

### 6.59.4.3 InstructionHandle **umbra.instructions.ast.InstructionLineController.my\_instr\_handle** [private]

A BCEL handle to the current instruction representation in BCEL format.

Referenced by `umbra.instructions.ast.InstructionLineController.getHandle()`, and `umbra.instructions.ast.InstructionLineController.replace()`.

### 6.59.4.4 MethodGen **umbra.instructions.ast.InstructionLineController.my\_methodgen** [private]

A BCEL object that represents the method in which the current instruction is located.

Referenced by `umbra.instructions.ast.InstructionLineController.getMethod()`.

### 6.59.4.5 String **umbra.instructions.ast.InstructionLineController.my\_name** [private]

The mnemonic name of the current instruction.

Referenced by `umbra.instructions.ast.InstructionLineController.getName()`, and `umbra.instructions.ast.InstructionLineController.setName()`.

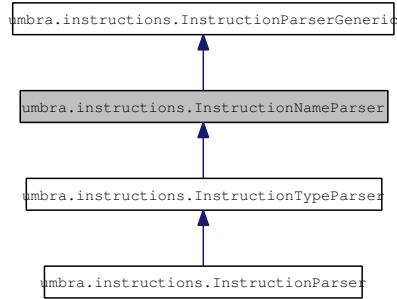
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[InstructionLineController.java](#)

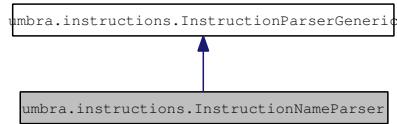
## 6.60 umbra.instructions.InstructionNameParser Class Reference

This class is the part of the byte code instruction parser which contributes the parsing of various identifiers i.e.

Inheritance diagram for umbra.instructions.InstructionNameParser:



Collaboration diagram for umbra.instructions.InstructionNameParser:



### Public Member Functions

- boolean [swallowClassname \(\)](#)

*This method swallows a single class name.*

- boolean [swallowFieldName \(\)](#)

*This method swallows a single field name with different possible name chunk separators.*

### Protected Member Functions

- [InstructionNameParser \(final String a\\_line\)](#)

*This constructor sets the string to be parsed and resets the parser so that it is ready to analyse the content.*

- boolean [swallowClassnameWithDelim \(final char a\\_separator\)](#)

*This method swallows a single class name with different possible name chunk separators.*

- boolean [swallowMethodName \(\)](#)

*This method swallows a single method name.*

## Private Member Functions

- boolean `swallowIdentifier()`  
*This method swallows a single proper identifier.*

### 6.60.1 Detailed Description

This class is the part of the byte code instruction parser which contributes the parsing of various identifiers i.e.

field names, class names, and method names.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.60.2 Constructor & Destructor Documentation

#### 6.60.2.1 `umbra.instructions.InstructionNameParser.InstructionNameParser (final String a_line)` [protected]

This constructor sets the string to be parsed and resets the parser so that it is ready to analyse the content. It relies on the work in the superclass. It can be called only from subclasses.

#### Parameters:

`a_line` the line with the content to parse

### 6.60.3 Member Function Documentation

#### 6.60.3.1 `boolean umbra.instructions.InstructionNameParser.swallowClassnameWithDelim (final char a_separator)` [protected]

This method swallows a single class name with different possible name chunk separators.

The separator is in the parameter `a_separator`. This method may not advance the index in case the first character to be analysed is not the proper first character of a class name. We assume the string is not finished before the method is called.

The Java class name (TypeName) is parsed using the following specification:

```
TypeName:  
    Identifier  
    TypeName separator Identifier
```

from JLS 3rd edition, 4.3 Reference Types and Values. We additionally assume that a Java classname is finished when it is followed either by whitespace or by one of '>', ';'.

**Parameters:**

*a\_separator* the name chunk separator

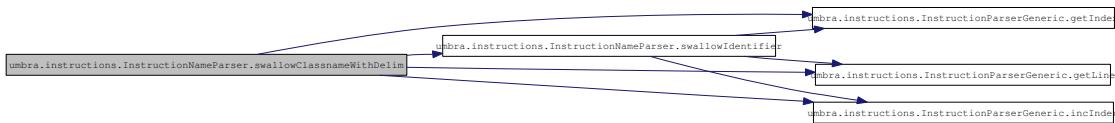
**Returns:**

true when the class name has been successfully swallowed, false otherwise.

References umbra.instructions.InstructionParserGeneric.getIndex(), umbra.instructions.InstructionParserGeneric.getLine(), umbra.instructions.InstructionParserGeneric.incIndex(), and umbra.instructions.InstructionNameParser.swallowIdentifier().

Referenced by umbra.instructions.InstructionNameParser.swallowClassname(), and umbra.instructions.InstructionTypeParser.swallowObjectTypeDescriptor().

Here is the call graph for this function:

**6.60.3.2 boolean umbra.instructions.InstructionNameParser.swallowClassname ()**

This method swallows a single class name.

This method may not advance the index in case the first character to be analysed is not the proper first character of a class name. We assume the string is not finished before the method is called.

The Java class name (TypeName) is parsed using the following specification:

```

TypeName:
  Identifier
  TypeName . Identifier

```

from JLS 3rd edition, 4.3 Reference Types and Values. We additionally assume that a Java classname is finished when it is followed either by whitespace or by '>'.

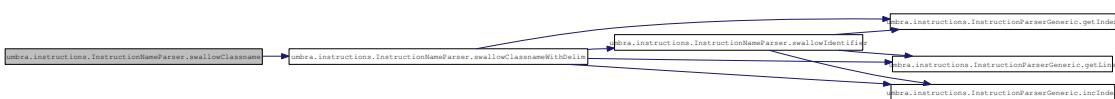
**Returns:**

true when the class name has been successfully swallowed, false otherwise.

References umbra.instructions.InstructionNameParser.swallowClassnameWithDelim().

Referenced by umbra.instructions.ast.NewInstruction.classnameWithDelimiters().

Here is the call graph for this function:



### 6.60.3.3 boolean umbra.instructions.InstructionNameParser.swallowIdentifier () [private]

This method swallows a single proper identifier.

This method may not advance the index in case the first character to be analysed is not the proper first character of an identifier. We assume the string is not finished before the method is called.

The exact format, according to JLS 3rd edition 3.8 Identifiers, is:

```
Identifier:
IdentifierChars but not a Keyword or BooleanLiteral or NullLiteral

IdentifierChars:
JavaLetter
IdentifierChars JavaLetterOrDigit
```

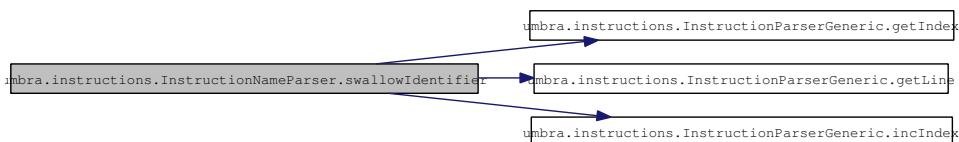
where a "JavaLetter" is a character for which the method [Character#isJavaIdentifierStart\(int\)](#) returns true and a "JavaLetterOrDigit" is a character for which the method [Character#isJavaIdentifierPart\(int\)](#) returns true.

#### Returns:

`true` when the identifier has been properly identified and swallowed, `false` when the starting portion of the string cannot start an identifier

References                    `umbra.instructions.InstructionParserGeneric.getIndex()`,                    um-  
`bra.instructions.InstructionParserGeneric.getLine()`, and `umbra.instructions.InstructionParserGeneric.incIndex()`.  
Referenced        by        `umbra.instructions.InstructionNameParser.swallowClassnameWithDelim()`,  
`umbra.instructions.InstructionNameParser.swallowFieldName()`,                    and                    um-  
`bra.instructions.InstructionNameParser.swallowMethodName()`.

Here is the call graph for this function:



### 6.60.3.4 boolean umbra.instructions.InstructionNameParser.swallowFieldName ()

This method swallows a single field name with different possible name chunk separators.

The separator is in the parameter `a_separator`. This method may not advance the index in case the first character to be analysed is not the proper first character of a class name. We assume the string is not finished before the method is called.

We assume that a Java field name (TypeName) is parsed using the following specification:

```
FieldName:
Identifier
FieldName . Identifier
```

**FIXME:** this is not based on a part of JLS as I do not know where to find that; <https://mobius.ucd.ie/ticket/553>

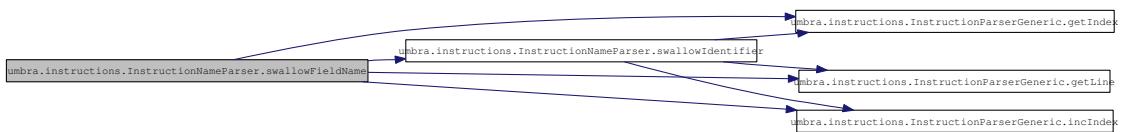
**Returns:**

`true` when the class name has been successfully swallowed, `false` otherwise.

References `umbra.instructions.InstructionParserGeneric.getIndex()`, `umbra.instructions.InstructionParserGeneric.getLine()`, `umbra.instructions.InstructionParserGeneric.incIndex()`, and `umbra.instructions.InstructionNameParser.swallowIdentifier()`.

Referenced by `umbra.instructions.ast.FieldInstruction.correct()`.

Here is the call graph for this function:



#### 6.60.3.5 boolean umbra.instructions.InstructionNameParser.swallowMethodName () [protected]

This method swallows a single method name.

This method may not advance the index in case the first character to be analysed is not the proper first character of a class name. We assume the string is not finished before the method is called.

The Java method name is parsed using the following specification:

```

MethodName:
  Identifier
  MethodName . Identifier

```

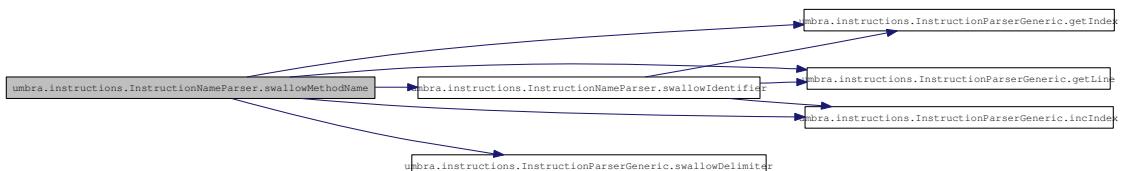
We additionally assume that a Java method is finished when it is followed by whitespace.

**Returns:**

`true` when the method name has been successfully swallowed, `false` otherwise.

References `umbra.instructions.InstructionParserGeneric.getIndex()`, `umbra.instructions.InstructionParserGeneric.getLine()`, `umbra.instructions.InstructionParserGeneric.incIndex()`, `umbra.instructions.InstructionParserGeneric.swallowDelimiter()`, and `umbra.instructions.InstructionNameParser.swallowIdentifier()`.

Here is the call graph for this function:



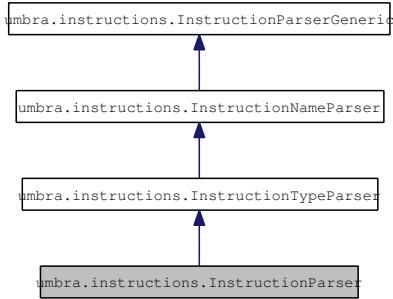
The documentation for this class was generated from the following file:

- source/umbra/instructions/[InstructionNameParser.java](#)

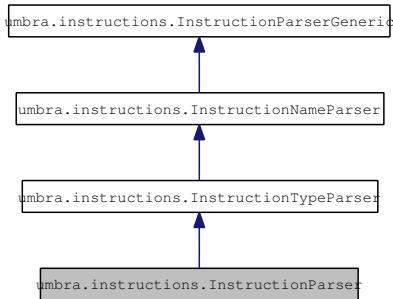
## 6.61 umbra.instructions.InstructionParser Class Reference

This class allows to parse the line with instruction.

Inheritance diagram for umbra.instructions.InstructionParser:



Collaboration diagram for umbra.instructions.InstructionParser:



### Public Member Functions

- **InstructionParser** (final String a\_line)
 

*This constructor sets the string to be parsed and resets the parser so that it is ready to analyse the content.*
- boolean **swallowNumber** ()
 

*This method swallows all the digits starting from the current position of the index.*
- int **swallowMnemonic** (final String[ ] the\_inventory)
 

*Checks if the line at the current position starts with a mnemonic from the inventory.*
- int **getResult** ()

### Private Attributes

- int **my\_result**

*This field contains the number parsed from the chunk of the digits.*
- int **my\_mnemonicno** = -1
 

*The number of the last parsed mnemonic.*

### 6.61.1 Detailed Description

This class allows to parse the line with instruction.

It enables the analysis of the correctness.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.61.2 Constructor & Destructor Documentation

#### 6.61.2.1 umbra.instructions.InstructionParser.InstructionParser (final String *a\_line*)

This constructor sets the string to be parsed and resets the parser so that it is ready to analyse the content.

It relies on the work in the superclass.

**Parameters:**

*a\_line* the line with the content to parse

### 6.61.3 Member Function Documentation

#### 6.61.3.1 boolean umbra.instructions.InstructionParser.swallowNumber ()

This method swallows all the digits starting from the current position of the index.

This method may not advance the index in case the first character to be analysed is not a digit or the analysis is finished before the method is called. This method assumes that a number is finished when a whitespace or end of string is met. In case the whitespace is not met after the string the number is not considered to be successfully swallowed.

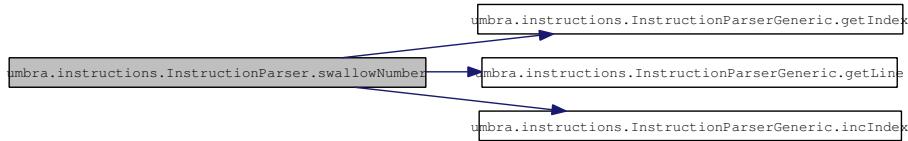
**Returns:**

true when a number was successfully swallowed, false otherwise

References                   umbra.instructions.InstructionParserGeneric.getIndex(),                   um-  
                               bra.instructions.InstructionParserGeneric.getLine(),                   umbra.instructions.InstructionParserGeneric.incIndex(),  
                               and                           umbra.instructions.InstructionParser.my\_result.

Referenced       by       umbra.instructions.ast.NumInstruction.checkNoParameters(),                   um-  
                       bra.instructions.ast.StackInstruction.correct(),                   umbra.instructions.ast.PushInstruction.correct(),                   um-  
                       bra.instructions.ast.LdcInstruction.correct(),                   umbra.instructions.ast.JumpInstruction.correct(),                   um-  
                       bra.instructions.ast.IincInstruction.correct(),                   umbra.instructions.ast.StackInstruction.getInd(),                   um-  
                       bra.instructions.ast.PushInstruction.getInd(),                   umbra.instructions.ast.JumpInstruction.getInd(),                   um-  
                       bra.instructions.ast.IincInstruction.getInd1(),                   umbra.instructions.ast.IincInstruction.getInd2(),                   um-  
                       bra.instructions.ast.InvokeInstruction.invokeinterfaceParams(),                   umbra.instructions.ast.MultiInstruction.numberWithDelimiters()  
                               and                           umbra.instructions.ast.InstructionLineController.parseTillMnemonic().

Here is the call graph for this function:



### 6.61.3.2 int umbra.instructions.InstructionParser.swallowMnemonic (final String[ ] the\_inventory)

Checks if the line at the current position starts with a mnemonic from the inventory.

**Parameters:**

`the_inventory` the array of the mnemonics to be checked

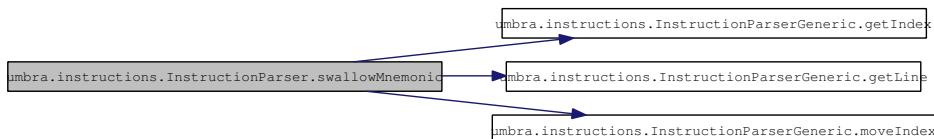
**Returns:**

the index to the entry in the inventory which contains the mnemonic or -1 in case no mnemonic from the inventory occurs

References `umbra.instructions.InstructionParserGeneric.getIndex()`, `umbra.instructions.InstructionParserGeneric.getLine()`, `umbra.instructions.InstructionParserGeneric.moveIndex()`, and `umbra.instructions.InstructionParser.my_mnemonicno`.

Referenced by `umbra.instructions.ast.NumInstruction.checkInstructionWithNumber()`, `umbra.instructions.ast.StackInstruction.correct()`, `umbra.instructions.ast.PushInstruction.correct()`, `umbra.instructions.ast.NewInstruction.correct()`, `umbra.instructions.ast.LdcInstruction.correct()`, `umbra.instructions.ast.JumpInstruction.correct()`, `umbra.instructions.ast.InvokeInstruction.correct()`, `umbra.instructions.ast.IincInstruction.correct()`, `umbra.instructions.ast.FieldInstruction.correct()`, `umbra.instructions.ast.ArrayInstruction.correct()`, `umbra.instructions.ast.IincInstruction.getInd1()`, and `umbra.instructions.ast.ArrayInstruction.getType()`.

Here is the call graph for this function:



### 6.61.3.3 int umbra.instructions.InstructionParser.getResult ()

**Returns:**

the number which is the result of parsing

References `umbra.instructions.InstructionParser.my_result`.

Referenced by `umbra.instructions.ast.StackInstruction.getInd()`, `umbra.instructions.ast.PushInstruction.getInd()`, `umbra.instructions.ast.JumpInstruction.getInd()`, `umbra.instructions.ast.IincInstruction.getInd1()`, `umbra.instructions.ast.IincInstruction.getInd2()`, and `umbra.instructions.ast.InvokeInstruction.invokeInterfaceParams()`.

## 6.61.4 Member Data Documentation

### 6.61.4.1 int umbra.instructions.InstructionParser.my\_result [private]

This field contains the number parsed from the chunk of the digits.

It contains a sensible value right after the [swallowNumber\(\)](#) is called.

Referenced by [umbra.instructions.InstructionParser.getResult\(\)](#), and [umbra.instructions.InstructionParser.swallowNumber\(\)](#).

### 6.61.4.2 int umbra.instructions.InstructionParser.my\_mnemonicno = -1 [private]

The number of the last parsed mnemonic.

The number is an index in the array given as the parameter to [swallowMnemonic\(String\[ \]\)](#). If no sensible mnemonic have been found the field has the value -1;

Referenced by [umbra.instructions.InstructionParser.swallowMnemonic\(\)](#).

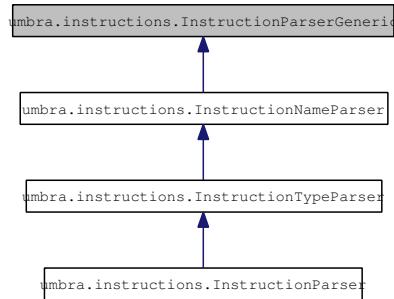
The documentation for this class was generated from the following file:

- source/umbra/instructions/[InstructionParser.java](#)

## 6.62 umbra.instructions.InstructionParserGeneric Class Reference

This class is the initial part of the byte code instruction parser class.

Inheritance diagram for umbra.instructions.InstructionParserGeneric:



### Public Member Functions

- boolean [swallowWhitespace \(\)](#)

*This method swallows all the whitespace starting from the current position of the index.*

- boolean [swallowDelimiter \(final char a\\_ch\)](#)

*This method swallows the given delimiter.*

- String [getLine \(\)](#)

*Returns the content of the line which is parsed.*

- int [getIndex \(\)](#)

*Returns the current index in the parsed line.*

- void [resetParser \(\)](#)

*This method resets the parser so that it starts the analysis from the beginning.*

- int [incIndex \(\)](#)

*This method moves the index inside the parser one position forward.*

- int [moveIndex \(final int a\\_step\)](#)

*This method moves the index inside the parser the given number of positions forward.*

- boolean [isFinished \(\)](#)

### Protected Member Functions

- [InstructionParserGeneric \(final String a\\_line\)](#)

*This constructor sets the string to be parsed and resets the parser so that it is ready to analyse the content.*

## Private Attributes

- String `my_line`  
*This field contains the value of the instruction line which is parsed.*
- int `my_index`  
*The pointer inside the line.*

### 6.62.1 Detailed Description

This class is the initial part of the byte code instruction parser class.

This parser is constructed as a sequence of subclasses that define various parts of the parser functionality. This class contains the most basic operations e.g. to swallow whitespace characters or to swallow delimiter

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.62.2 Constructor & Destructor Documentation

#### 6.62.2.1 umbra.instructions.InstructionParserGeneric.InstructionParserGeneric (final String `a_line`) [protected]

This constructor sets the string to be parsed and resets the parser so that it is ready to analyse the content.

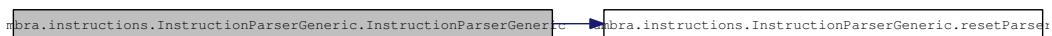
The constructor can only be called from subclasses.

#### Parameters:

`a_line` the line with the content to parse

References      `umbra.instructions.InstructionParserGeneric.my_line`,      and      `umbra.instructions.InstructionParserGeneric.resetParser()`.

Here is the call graph for this function:



### 6.62.3 Member Function Documentation

#### 6.62.3.1 boolean umbra.instructions.InstructionParserGeneric.swallowWhitespace ()

This method swallows all the whitespace starting from the current position of the index.

This method may not advance the index in case the first character to be analysed is not whitespace.

**Returns:**

`true` when the further analysis is not finished yet, `false` when at the end of the string

References                `umbra.instructions.InstructionParserGeneric.my_index`,                and  
`umbra.instructions.InstructionParserGeneric.my_line`.

Referenced by `umbra.instructions.ast.NumInstruction.checkInstructionWithNumber()`, `umbra.instructions.ast.NumInstruction.checkNoParameters()`, `umbra.instructions.ast.NewInstruction.classnameWithDelimiters()`, `umbra.instructions.ast.StackInstruction.correct()`, `umbra.instructions.ast.PushInstruction.correct()`, `umbra.instructions.ast.NewInstruction.correct()`, `umbra.instructions.ast.LdcInstruction.correct()`, `umbra.instructions.ast.JumpInstruction.correct()`, `umbra.instructions.ast.InvokeInstruction.correct()`, `umbra.instructions.ast.IincInstruction.correct()`, `umbra.instructions.ast.FieldInstruction.correct()`, `umbra.instructions.ast.ArrayInstruction.correct()`, `umbra.instructions.ast.StackInstruction.getInd()`, `umbra.instructions.ast.PushInstruction.getInd()`, `umbra.instructions.ast.IincInstruction.getInd1()`, `umbra.instructions.ast.IincInstruction.getInd2()`, `umbra.instructions.ast.ArrayInstruction.getType()`, `umbra.instructions.ast.InvokeInstruction.invokeInterfaceParams()`, `umbra.instructions.ast.MultiInstruction.numberWithDelimiters()`, `umbra.instructions.ast.InstructionLineController.parseTillMnemonic()` and `umbra.instructions.ast.LdcInstruction.stringWithDelimiters()`.

**6.62.3.2 boolean umbra.instructions.InstructionParserGeneric.swallowDelimiter (final char *a\_ch*)**

This method swallows the given delimiter.

This method may not advance the index in case the first character to be analysed is not the delimiter or the analysis is finished before the method is called.

**Parameters:**

*a\_ch* the character with the delimiter to swallow

**Returns:**

`true` when the delimiter was successfully swallowed, `false` otherwise

References                `umbra.instructions.InstructionParserGeneric.my_index`,                and  
`umbra.instructions.InstructionParserGeneric.my_line`.

Referenced by `umbra.instructions.ast.NumInstruction.checkInstructionWithNumber()`, `umbra.instructions.ast.NewInstruction.classnameWithDelimiters()`, `umbra.instructions.ast.StackInstruction.correct()`, `umbra.instructions.ast.JumpInstruction.correct()`, `umbra.instructions.ast.IincInstruction.correct()`, `umbra.instructions.ast.ArrayInstruction.correct()`, `umbra.instructions.ast.JumpInstruction.getInd()`, `umbra.instructions.ast.IincInstruction.getInd1()`, `umbra.instructions.ast.MultiInstruction.numberWithDelimiters()`, `umbra.instructions.ast.InstructionLineController.parseTillMnemonic()`, `umbra.instructions.ast.LdcInstruction.stringWithDelimiters()`, `umbra.instructions.InstructionNameParser.swallowMethodName()` and `umbra.instructions.InstructionTypeParser.swallowObjectTypeDescriptor()`.

**6.62.3.3 String umbra.instructions.InstructionParserGeneric.getLine ()**

Returns the content of the line which is parsed.

**Returns:**

the content of the current line

References umbra.instructions.InstructionParserGeneric.my\_line.

Referenced by umbra.instructions.ast.LdcInstruction.correct(), umbra.instructions.InstructionNameParser.swallowClassnameWithDelim(), umbra.instructions.InstructionNameParser.swallowFieldName(), umbra.instructions.InstructionTypeParser.swallowFieldType(), umbra.instructions.InstructionNameParser.swallowIdentifier(), umbra.instructions.InstructionNameParser.swallowMethodNames(), umbra.instructions.InstructionParser.swallowMnemonic(), umbra.instructions.InstructionParser.swallowNumber(), and umbra.instructions.InstructionTypeParser.swallowRefTypeDescriptor().

#### 6.62.3.4 int umbra.instructions.InstructionParserGeneric.getIndex ()

Returns the current index in the parsed line.

**Returns:**

the index in the current line

References umbra.instructions.InstructionParserGeneric.my\_index.

Referenced by umbra.instructions.ast.LdcInstruction.correct(), umbra.instructions.InstructionNameParser.swallowClassnameWithDelim(), umbra.instructions.InstructionNameParser.swallowFieldName(), umbra.instructions.InstructionTypeParser.swallowFieldType(), umbra.instructions.InstructionNameParser.swallowIdentifier(), umbra.instructions.InstructionNameParser.swallowMethodNames(), umbra.instructions.InstructionParser.swallowMnemonic(), umbra.instructions.InstructionParser.swallowNumber(), and umbra.instructions.InstructionTypeParser.swallowRefTypeDescriptor().

#### 6.62.3.5 void umbra.instructions.InstructionParserGeneric.resetParser ()

This method resets the parser so that it starts the analysis from the beginning.

References umbra.instructions.InstructionParserGeneric.my\_index.

Referenced by umbra.instructions.ast.StackInstruction.getInd(), umbra.instructions.ast.PushInstruction.getInd(), umbra.instructions.ast.JumpInstruction.getInd(), umbra.instructions.ast.ArrayInstruction.getType(), umbra.instructions.InstructionParserGeneric.InstructionParserGeneric(), and umbra.instructions.ast.InstructionLineController.parseTillMnemonic().

#### 6.62.3.6 int umbra.instructions.InstructionParserGeneric.incIndex ()

This method moves the index inside the parser one position forward.

**Returns:**

the new value of the index

References umbra.instructions.InstructionParserGeneric.my\_index.

Referenced by umbra.instructions.InstructionNameParser.swallowClassnameWithDelim(), umbra.instructions.InstructionNameParser.swallowFieldName(), umbra.instructions.InstructionTypeParser.swallowFieldType(), umbra.instructions.InstructionNameParser.swallowIdentifier(), umbra.instructions.InstructionNameParser.swallowMethodNames(), umbra.instructions.InstructionParser.swallowNumber(), and umbra.instructions.InstructionTypeParser.swallowRefTypeDescriptor().

#### 6.62.3.7 int umbra.instructions.InstructionParserGeneric.moveIndex (final int *a\_step*)

This method moves the index inside the parser the given number of positions forward.

**Parameters:**

*a\_step* a number by which the index is advanced

**Returns:**

the new value of the index

References `umbra.instructions.InstructionParserGeneric.my_index`.

Referenced by `umbra.instructions.InstructionParser.swallowMnemonic()`.

**6.62.3.8 boolean umbra.instructions.InstructionParserGeneric.isFinished ()****Returns:**

`true` when the index is at the end of the parsed string

References `umbra.instructions.InstructionParserGeneric.my_index`, and `umbra.instructions.InstructionParserGeneric.my_line`.

**6.62.4 Member Data Documentation****6.62.4.1 String umbra.instructions.InstructionParserGeneric.my\_line [private]**

This field contains the value of the instruction line which is parsed.

Referenced by `umbra.instructions.InstructionParserGeneric.getLine()`, `umbra.instructions.InstructionParserGeneric InstructionParserGeneric()`, `umbra.instructions.InstructionParserGeneric.isFinished()`, `umbra.instructions.InstructionParserGeneric.swallowDelimiter()`, and `umbra.instructions.InstructionParserGeneric.swallowWhitespace()`.

**6.62.4.2 int umbra.instructions.InstructionParserGeneric.my\_index [private]**

The pointer inside the line.

It points to the first character which has not been analysed yet. If this field is equal to `my_line.length()` the analysis is finished.

Referenced by `umbra.instructions.InstructionParserGeneric.getIndex()`, `umbra.instructions.InstructionParserGeneric.incIndex()`, `umbra.instructions.InstructionParserGeneric.isFinished()`, `umbra.instructions.InstructionParserGeneric.moveIndex()`, `umbra.instructions.InstructionParserGeneric.resetParser()`, `umbra.instructions.InstructionParserGeneric.swallowDelimiter()`, and `umbra.instructions.InstructionParserGeneric.swallowWhitespace()`.

The documentation for this class was generated from the following file:

- `source/umbra/instructions/`[InstructionParserGeneric.java](#)

## 6.63 umbra.instructions.InstructionParserHelper Class Reference

This class contains helper methods that allow check the classes of various syntactical classes occurring in Java byte code files.

### Static Public Member Functions

- static String [getEOL](#) ()  
*Checks if the given character starts an array type descriptor.*
- static boolean [isArrayTypeDescriptor](#) (final char a\_c)  
*Checks if the given character starts a base type descriptor.*
- static boolean [isBaseTypeDescriptor](#) (final char a\_c)  
*Check if a character is a meaningful escape character.*
- static boolean [isJavaKeyword](#) (finalString a\_string)  
*Checks if the given string is a Java keyword.*
- static boolean [isJavaResLiteral](#) (finalString a\_string)  
*Checks if the given string is a Java reserved literal.*
- static boolean [isObjectTypeDescriptor](#) (final char a\_c)  
*Checks if the given character starts an object type descriptor.*
- static boolean [isOctalDigit](#) (final char a\_char)  
*Check if a character is an octal digit.*
- static boolean [isVoidTypeDescriptor](#) (final char a\_c)  
*Checks if the given character starts a void type descriptor.*
- static boolean [isZeroToThreeDigit](#) (final char a\_char)  
*Check if a character is 0, 1, 2, or 3.*

### Static Public Attributes

- static final int [MAX\\_OCTAL\\_NUMBER\\_LENGTH](#) = 3  
*The maximal length of an octal escape.*

### Private Member Functions

- [InstructionParserHelper](#) ()  
*Empty private constructor to forbid the creation of objects with this type.*

## Static Private Attributes

- static final String[ ] **JAVA\_RES\_LITERALS**  
*Java reserved literals as enumerated in JLS 3rd edition, 3.9 Keywords.*
- static final String[ ] **JAVA\_KEYWORDS**  
*Java reserved keywords as enumerated in JLS 3rd edition, 3.9 Keywords.*
- static final String **OCTAL\_DIGITS** = "01234567"  
*Octal digits.*
- static final String **ZEROTOTHREE\_DIGITS** = "0123"  
*Zero-three digits.*
- static final String **BASE\_TYPE\_DESCRIPTOR** = "BCDFIJSZ"  
*Base type descriptor characters.*
- static final String **ESCAPE\_CODE\_CHARACTERS** = "btnfr\\'\\\\\"  
*The meaningful escape characters.*
- static String **a\_LINE\_SEPARATOR**  
*Contains cached line separator string.*

### 6.63.1 Detailed Description

This class contains helper methods that allow check the classes of various syntactical classes occurring in Java byte code files.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.63.2 Constructor & Destructor Documentation

#### 6.63.2.1 **umbra.instructions.InstructionParserHelper.InstructionParserHelper ()** [private]

Empty private constructor to forbid the creation of objects with this type.

### 6.63.3 Member Function Documentation

#### 6.63.3.1 **static String umbra.instructions.InstructionParserHelper.getEOL ()** [static]

#### Returns:

the line separator specific for the current system

References `umbra.instructions.InstructionParserHelper.a_LINE_SEPARATOR`.

**6.63.3.2 static boolean umbra.instructions.InstructionParserHelper.isArrayTypeDescriptor (final char *a\_c*) [static]**

Checks if the given character starts an array type descriptor.

**Parameters:**

*a\_c* a character to check

**Returns:**

true when the character starts an array type descriptor

**6.63.3.3 static boolean umbra.instructions.InstructionParserHelper.isBaseTypeDescriptor (final char *a\_c*) [static]**

Checks if the given character starts a base type descriptor.

**Parameters:**

*a\_c* a character to check

**Returns:**

true when the character starts a byse type descriptor

References umbra.instructions.InstructionParserHelper.BASE\_TYPE\_DESCRIPTORs.

**6.63.3.4 static boolean umbra.instructions.InstructionParserHelper.isEscapeChar (final char *a\_char*) [static]**

Check if a character is a meaningful escape character.

The meaningful escape characters are as described in JLS 3rd edition, 3.10.6 Escape Sequences for Character and String Literals: b, t, n, f, r, ", ', \.

**Parameters:**

*a\_char* the character to check

**Returns:**

true when a\_char is a meaningful escape character

References umbra.instructions.InstructionParserHelper.ESCAPE\_CODE\_CHARACTERS.

**6.63.3.5 static boolean umbra.instructions.InstructionParserHelper.isJavaKeyword (final String *a\_string*) [static]**

Checks if the given string is a Java keyword.

**Parameters:**

*a\_string* the string to check

**Returns:**

true when the given string is a Java keyword, false otherwise

References umbra.instructions.InstructionParserHelper.JAVA\_KEYWORDS.

**6.63.3.6 static boolean umbra.instructions.InstructionParserHelper.isJavaResLiteral (final String a\_string) [static]**

Checks if the given string is a Java reserved literal.

**Parameters:**

*a\_string* the string to check

**Returns:**

true when the given string is a Java reserved literal false otherwise

References umbra.instructions.InstructionParserHelper.JAVA\_RES\_LITERALS.

**6.63.3.7 static boolean umbra.instructions.InstructionParserHelper.isObjectTypeDescriptor (final char a\_c) [static]**

Checks if the given character starts an object type descriptor.

**Parameters:**

*a\_c* a character to check

**Returns:**

true when the character starts an object type descriptor

**6.63.3.8 static boolean umbra.instructions.InstructionParserHelper.isOctalDigit (final char a\_char) [static]**

Check if a character is an octal digit.

**Parameters:**

*a\_char* the character to check

**Returns:**

true when a\_char is an octal digit

References umbra.instructions.InstructionParserHelper.OCTAL\_DIGITS.

**6.63.3.9 static boolean umbra.instructions.InstructionParserHelper.isVoidTypeDescriptor (final char *a\_c*) [static]**

Checks if the given character starts a void type descriptor.

**Parameters:**

*a\_c* a character to check

**Returns:**

true when the character starts a void type descriptor

**6.63.3.10 static boolean umbra.instructions.InstructionParserHelper.isZeroToThreeDigit (final char *a\_char*) [static]**

Check if a character is 0, 1, 2, or 3.

**Parameters:**

*a\_char* the character to check

**Returns:**

true when *a\_char* is 0, 1, 2, or 3

References umbra.instructions.InstructionParserHelper.ZEROTOTHREE\_DIGITS.

## 6.63.4 Member Data Documentation

**6.63.4.1 final int umbra.instructions.InstructionParserHelper.MAX\_OCTAL\_NUMBER\_LENGTH = 3 [static]**

The maximal length of an octal escape.

**6.63.4.2 final String [] umbra.instructions.InstructionParserHelper.JAVA\_RES\_LITERALS [static, private]****Initial value:**

```
{  
    "null",  "true",   "false"  
}
```

Java reserved literals as enumerated in JLS 3rd edition, 3.9 Keywords.

Referenced by umbra.instructions.InstructionParserHelper.isJavaResLiteral().

**6.63.4.3 final String [] umbra.instructions.InstructionParserHelper.JAVA\_KEYWORDS [static, private]****Initial value:**

```
{
    "abstract",   "continue",      "for",          "new",           "switch",
    "assert",     "default",       "if",           "package",       "synchronized",
    "boolean",    "do",            "goto",         "private",        "this",
    "break",      "double",        "implements",  "protected",     "throw",
    "byte",       "else",          "import",       "public",        "throws",
    "case",       "enum",          "instanceof",  "return",        "transient",
    "catch",      "extends",        "int",          "short",         "try",
    "char",       "final",          "interface",   "static",        "void",
    "class",      "finally",        "long",         "strictfp",     "volatile",
    "const",      "float",          "native",       "super",         "while"
}
```

Java reserved keywords as enumerated in JLS 3rd edition, 3.9 Keywords.

Referenced by `umbra.instructions.InstructionParserHelper.isJavaKeyword()`.

#### **6.63.4.4 final String umbra.instructions.InstructionParserHelper.OCTAL\_DIGITS = "01234567"** [static, private]

Octal digits.

Referenced by `umbra.instructions.InstructionParserHelper.isOctalDigit()`.

#### **6.63.4.5 final String umbra.instructions.InstructionParserHelper.ZEROTOTHREE\_DIGITS = "0123"** [static, private]

Zero-three digits.

Referenced by `umbra.instructions.InstructionParserHelper.isZeroToThreeDigit()`.

#### **6.63.4.6 final String umbra.instructions.InstructionParserHelper.BASE\_TYPE\_DESCRIPTOR = "BCDFIJSZ"** [static, private]

Base type descriptor characters.

Referenced by `umbra.instructions.InstructionParserHelper.isBaseTypeDescriptor()`.

#### **6.63.4.7 final String umbra.instructions.InstructionParserHelper.ESCAPE\_CODE\_CHARACTERS = "btnfr\\\"\\\\"** [static, private]

The meaningful escape characters.

These are as described in JLS 3rd edition, 3.10.6 Escape Sequences for Character and String Literals: b, t, n, f, r, ", ', \.

Referenced by `umbra.instructions.InstructionParserHelper.isEscapeChar()`.

#### **6.63.4.8 String umbra.instructions.InstructionParserHelper.a\_LINE\_SEPARATOR [static, private]**

Contains cached line separator string.

Referenced by `umbra.instructions.InstructionParserHelper.getEOL()`.

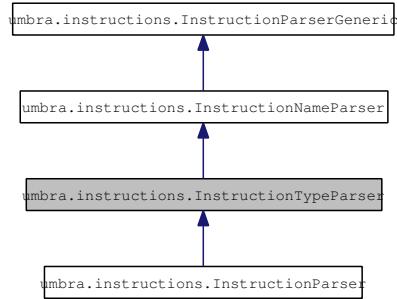
The documentation for this class was generated from the following file:

- source/umbra/instructions/[InstructionParserHelper.java](#)

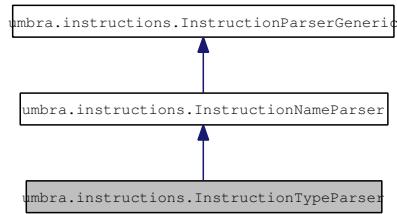
## 6.64 umbra.instructions.InstructionTypeParser Class Reference

This class is the part of the byte code instruction parser which contributes the parsing of various type representations.

Inheritance diagram for umbra.instructions.InstructionTypeParser:



Collaboration diagram for umbra.instructions.InstructionTypeParser:



### Public Member Functions

- boolean [swallowFieldType \(\)](#)

*This method swallows a filed type descriptor.*

### Protected Member Functions

- [InstructionTypeParser \(final String a\\_line\)](#)

*This constructor sets the string to be parsed and resets the parser so that it is ready to analyse the content.*

- boolean [swallowRefTypeDescriptor \(\)](#)

*This method swallows a reference type descriptor.*

### Private Member Functions

- boolean [swallowArrayTypeDescriptor \(\)](#)

*This method swallows an array type descriptor.*

- boolean [swallowObjectTypeDescriptor \(\)](#)

*This method swallows an object type descriptor.*

### 6.64.1 Detailed Description

This class is the part of the byte code instruction parser which contributes the parsing of various type representations.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.64.2 Constructor & Destructor Documentation

#### 6.64.2.1 umbra.instructions.InstructionTypeParser.InstructionTypeParser (final String *a\_line*) [protected]

This constructor sets the string to be parsed and resets the parser so that it is ready to analyse the content. It relies on the work in the superclass.

#### Parameters:

*a\_line* the line with the content to parse

### 6.64.3 Member Function Documentation

#### 6.64.3.1 boolean umbra.instructions.InstructionTypeParser.swallowFieldType ()

This method swallows a filed type descriptor.

This method may not advance the index in case the first character to be analysed is not the proper first character of an array descriptor. We assume the string is not finished before the method is called.

As JVMS, 4.3.3 Method Descriptors says, a filed type descriptor is a series of characters generated by the grammar:

```
FileType:  
  BaseType  
  ArrayType  
  ObjectType ;
```

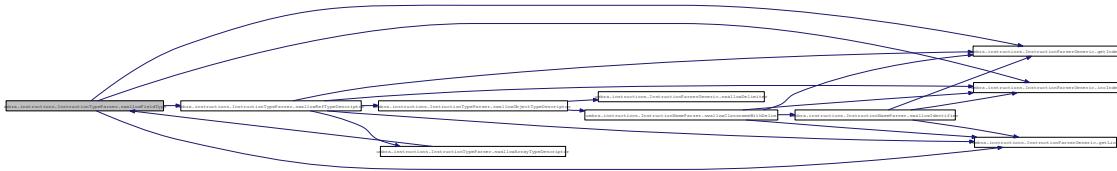
#### Returns:

`true` when a return descriptor is successfully swallowed, `false` otherwise

References                  [umbra.instructions.InstructionParserGeneric.getIndex\(\)](#),                  [umbra.instructions.InstructionParserGeneric.getLine\(\)](#), [umbra.instructions.InstructionParserGeneric.incIndex\(\)](#),  
and [umbra.instructions.InstructionTypeParser.swallowRefTypeDescriptor\(\)](#).

Referenced by `umbra.instructions.ast.FieldInstruction.correct()`, and `umbra.instructions.InstructionTypeParser.swallowArrayTypeDescriptor()`.

Here is the call graph for this function:



#### 6.64.3.2 boolean `umbra.instructions.InstructionTypeParser.swallowRefTypeDescriptor ()` [protected]

This method swallows a reference type descriptor.

This method may not advance the index in case the first character to be analysed is not the proper first character of an array descriptor. We assume the string is not finished before the method is called.

As JVMS, 4.3.3 Method Descriptors says, a filed type descriptor is a series of characters generated by the grammar:

```
FiledType:
  BaseType
  ArrayType
  ObjectType ;
```

We omit here the `BaseType` case.

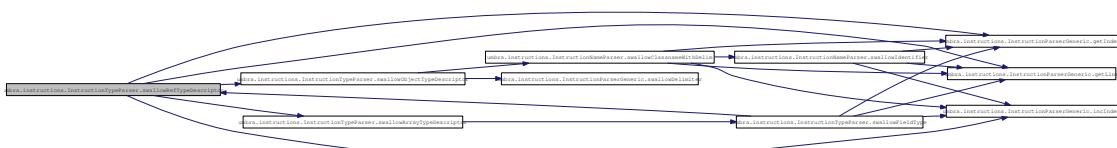
#### Returns:

`true` when a parameter descriptor is successfully swallowed, `false` otherwise

References `umbra.instructions.InstructionParserGeneric.getIndex()`, `umbra.instructions.InstructionParserGeneric.getLine()`, `umbra.instructions.InstructionParserGeneric.incIndex()`, `umbra.instructions.InstructionTypeParser.swallowArrayTypeDescriptor()`, and `umbra.instructions.InstructionTypeParser.swallowObjectTypeDescriptor()`.

Referenced by `umbra.instructions.InstructionTypeParser.swallowFieldType()`.

Here is the call graph for this function:



### 6.64.3.3 boolean umbra.instructions.InstructionTypeParser.swallowArrayTypeDescriptor () [private]

This method swallows an array type descriptor.

This method may not advance the index in case the first character to be analysed is not the proper first character of an array descriptor. We assume the string is not finished before the method is called.

As JVMS, 4.3.3 Method Descriptors says, an object type descriptor is a series of characters generated by the grammar:

```
ArrayType:  
[ ComponentType ;
```

we assume [ is already swallowed so we swallow here only the component type.

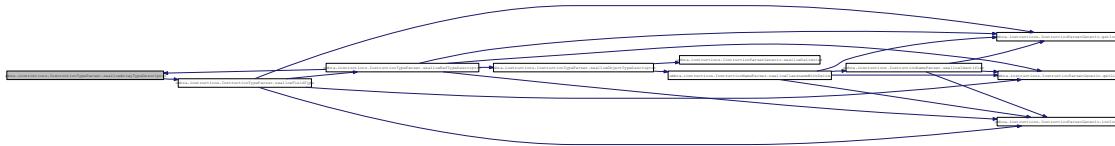
#### Returns:

`true` when a return descriptor is successfully swallowed, `false` otherwise

References `umbra.instructions.InstructionTypeParser.swallowFieldType()`.

Referenced by `umbra.instructions.InstructionTypeParser.swallowRefTypeDescriptor()`.

Here is the call graph for this function:



### 6.64.3.4 boolean umbra.instructions.InstructionTypeParser.swallowObjectTypeDescriptor () [private]

This method swallows an object type descriptor.

This method may not advance the index in case the first character to be analysed is not the proper first character of an object type descriptor. We assume the string is not finished before the method is called.

As JVMS, 4.3.3 Method Descriptors says, an object type descriptor is a series of characters generated by the grammar:

```
ObjectType:  
L <classname> ;
```

we assume L is already swallowed so we swallow here only the class name.

#### Returns:

`true` when a return descriptor is successfully swallowed, `false` otherwise

References `umbra.instructions.InstructionNameParser.swallowClassnameWithDelim()`, and `umbra.instructions.InstructionParserGeneric.swallowDelimiter()`.

Referenced by `umbra.instructions.InstructionTypeParser.swallowRefTypeDescriptor()`.

Here is the call graph for this function:



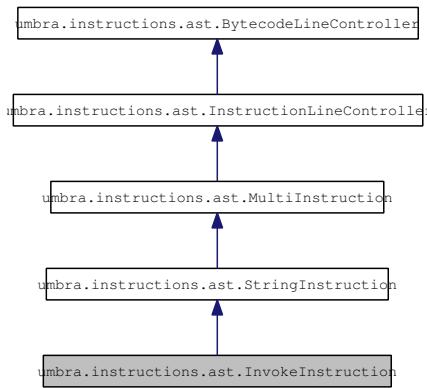
The documentation for this class was generated from the following file:

- source/umbra/instructions/[InstructionTypeParser.java](#)

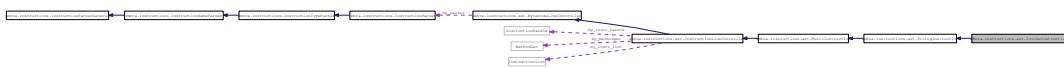
## 6.65 `umbra.instructions.ast.InvokeInstruction` Class Reference

This class handles the creation and correctness for invoke [instructions](#).

Inheritance diagram for `umbra.instructions.ast.InvokeInstruction`:



Collaboration diagram for `umbra.instructions.ast.InvokeInstruction`:



### Public Member Functions

- `InvokeInstruction (final String a_line_text, final String a_name)`

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.*

- `final boolean correct ()`

*Invoke instruction line is correct if its parameter contains class name at the beginning and a number in () at the end.*

- `final Instruction getInstruction ()`

*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for an invoke instruction.*

### Static Public Member Functions

- `static String[] getMnemonics ()`

*This method returns the array of invoke [instructions](#) mnemonics.*

### Private Member Functions

- `boolean invokeinterfaceParams (final InstructionParser a_parser)`

*This method tries to parse additional optional parameters of the invokeinterface instruction.*

### 6.65.1 Detailed Description

This class handles the creation and correctness for invoke [instructions](#).

The invoke [instructions](#) are:

- invokeinterface,
- invokespecial,
- invokestatic,
- invokevirtual.

**Author:**

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.65.2 Constructor & Destructor Documentation

#### 6.65.2.1 `umbra.instructions.ast.InvokeInstruction.InvokeInstruction (final String a_line_text, final String a_name)`

This creates an instance of an instruction named as *a\_name* in a line the text of which is *a\_line\_text*. Currently it just calls the constructor of the superclass.

**Parameters:**

- a\_line\_text* the line number of the instruction  
*a\_name* the mnemonic name of the instruction

**See also:**

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.65.3 Member Function Documentation

#### 6.65.3.1 `static String [ ] umbra.instructions.ast.InvokeInstruction.getMnemonics () [static]`

This method returns the array of invoke [instructions](#) mnemonics.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController](#).[getMnemonics](#)()

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

### 6.65.3.2 final boolean umbra.instructions.ast.InvokeInstruction.correct ()

Invoke instruction line is correct if its parameter contains class name at the beginning and a number in () at the end.

whitespase number : whitespace mnemonic whitespace whitespace ( whitespace number whitespace ) [ whitespace number whitespace number ] whitespace lineend where the text between [] is optional and occurs only when the mnemonic is "invokeinterface". Additionally the final number parameter should always be 0.

#### Returns:

`true` when the syntax of the instruction line is correct

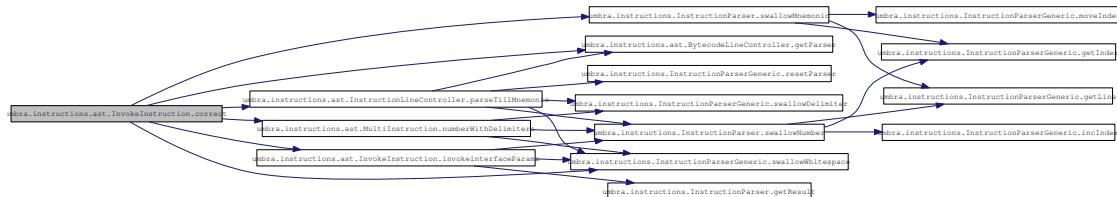
#### See also:

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

References [umbra.instructions.ast.BytocodeLineController.getParser\(\)](#), [umbra.instructions.ast.InvokeInstruction.invokeinterfaceParams\(\)](#), [umbra.instructions.ast.MultiInstruction.numberWithDelimiters\(\)](#), [umbra.instructions.ast.InstructionLineController.parseTillMnemonic\(\)](#), [umbra.instructions.InstructionParser.swallowMnemonic\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#).  
Referenced by [umbra.instructions.ast.InvokeInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



### 6.65.3.3 boolean umbra.instructions.ast.InvokeInstruction.invokeinterfaceParams (final InstructionParser *a\_parser*) [private]

This method tries to parse additional optional parameters of the invokeinterface instruction.

The precise format is: whitespace number whitespace number Additionally, the second number must be 0;

#### Parameters:

*a\_parser* the parser which is to parse the parameters

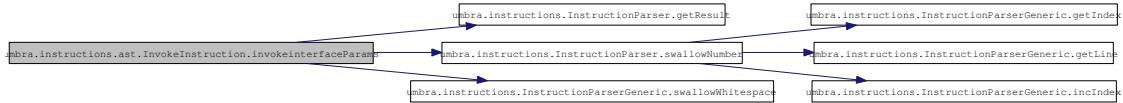
#### Returns:

`true` when the syntax of the parameters is correct

References [umbra.instructions.InstructionParser.getResult\(\)](#), [umbra.instructions.InstructionParser.swallowNumber\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#).

Referenced by [umbra.instructions.ast.InvokeInstruction.correct\(\)](#).

Here is the call graph for this function:



#### 6.65.3.4 final Instruction `umbra.instructions.ast.InvokeInstruction.getInstruction ()`

This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for an invoke instruction.

It computes the index parameter of the instruction before the instruction is constructed. The method can construct one of the [instructions](#):

- invokeinterface,
- invokespecial,
- invokestatic,
- invokevirtual.

This method also checks the syntactical correctness of the current instruction line.

#### Returns:

the freshly constructed BCEL instruction or `null` in case the instruction is not an invoke instruction and in case the line is incorrect

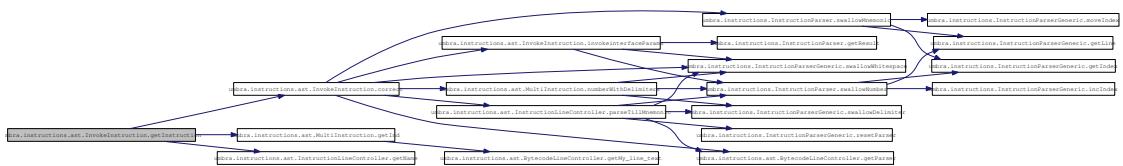
#### See also:

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References `umbra.instructions.ast.InvokeInstruction.correct()`, `umbra.instructions.ast.MultiInstruction.getInd()`, and `umbra.instructions.ast.InstructionLineController.getName()`.

Here is the call graph for this function:



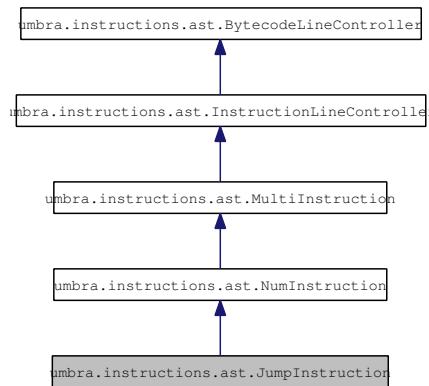
The documentation for this class was generated from the following file:

- `source/umbra/instructions/ast/InvokeInstruction.java`

## 6.66 umbra.instructions.ast.JumpInstruction Class Reference

This class handles the creation and correctness for jump [instructions](#).

Inheritance diagram for umbra.instructions.ast.JumpInstruction:



Collaboration diagram for umbra.instructions.ast.JumpInstruction:



### Public Member Functions

- [JumpInstruction](#) (final String a\_line\_text, final String a\_name)

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line.*

- final boolean [correct](#) ()

*Jump instruction line is correct if it has one simple number parameter preceded by '#'.*

- final Instruction [getInstruction](#) ()

*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for a jump instruction, i.e.*

- final void [setTarget](#) (final InstructionList an\_ins\_list, final Instruction an\_ins) throws UmbraException

*Jump instruction requires an instruction number of its target as a parameter.*

### Static Public Member Functions

- static String[] [getMnemonics](#) ()

*This method returns the array of jump [instructions](#) mnemonics.*

## Protected Member Functions

- int [getInd\(\)](#)  
*This method parses the parameter of the current instruction.*

## Private Member Functions

- Instruction [getSubroutineInstruction](#) (final InstructionHandle an\_ih, final Instruction a\_res)  
*This method creates the objects that represents a subroutine instruction.*
- Instruction [getNullCompIfInstruction](#) (final InstructionHandle an\_ih, final Instruction a\_res)  
*This method creates the objects that represent if [instructions](#) that compare with null references.*
- Instruction [getRefCompIfInstruction](#) (final InstructionHandle an\_ih, final Instruction a\_res)  
*This method creates the objects that represent if [instructions](#) that compare with references.*
- Instruction [getGotoInstruction](#) (final InstructionHandle an\_ih, final Instruction a\_res)  
*This method creates the objects that represent goto [instructions](#).*
- Instruction [getIntCompIfInstruction](#) (final InstructionHandle an\_ih, final Instruction a\_res)  
*This method creates the objects that represent if [instructions](#) to compare with integers.*
- Instruction [getZeroCompIfInstruction](#) (final InstructionHandle an\_ih, final Instruction a\_res)  
*This method creates the objects that represent if [instructions](#) that compare with integer 0.*

### 6.66.1 Detailed Description

This class handles the creation and correctness for jump [instructions](#).

The jump [instructions](#) are:

- unconditional goto [instructions](#),
- if [instructions](#) that compare references,
- if [instructions](#) that compare integers,
- if [instructions](#) that compare with null,
- if [instructions](#) that compare with 0,
- subroutine [instructions](#).

FIXME: "lookupswitch", "tableswitch" are handled in a special simplified way.  
<https://mobius.ucd.ie/ticket/552>

#### Author:

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksey Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

## 6.66.2 Constructor & Destructor Documentation

### 6.66.2.1 umbra.instructions.ast.JumpInstruction.JumpInstruction (final String *a\_line\_text*, final String *a\_name*)

This creates an instance of an instruction named as *a\_name* in a line the text of which is *a\_line*.

Currently it just calls the constructor of the superclass.

#### Parameters:

*a\_line\_text* the line number of the instruction

*a\_name* the mnemonic name of the instruction

#### See also:

[InstructionLineControllerInstructionLineController\(String, String\)](#)

## 6.66.3 Member Function Documentation

### 6.66.3.1 static String [ ] umbra.instructions.ast.JumpInstruction.getMnemonics () [static]

This method returns the array of jump [instructions](#) mnemonics.

#### Returns:

the array of the handled mnemonics

#### See also:

[InstructionLineController.getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

### 6.66.3.2 final boolean umbra.instructions.ast.JumpInstruction.correct ()

Jump instruction line is correct if it has one simple number parameter preceded by '#'.

The precise definition of this kind of a line is as follows: whitespace number : whitespace mnemonic whitespace # number whitespace lineend FIXME: tablesswitch and lookupswitch are handled in a special way <https://mobius.ucd.ie/ticket/552>

#### Returns:

true when the syntax of the instruction line is correct

#### See also:

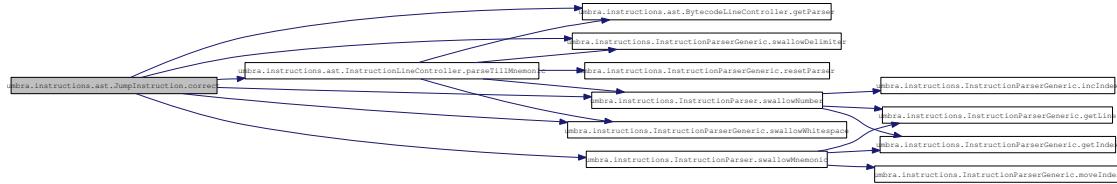
[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

References                [umbra.instructions.ast.BytocodeLineController.getParser\(\)](#),  
[umbra.instructions.ast.InstructionLineController.parseTillMnemonic\(\)](#),                um-  
[umbra.instructions.InstructionParserGeneric.swallowDelimiter\(\)](#), [umbra.instructions.InstructionParser.swallowMnemonic\(\)](#),                um-  
[umbra.instructions.InstructionParser.swallowNumber\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#)

Referenced by `umbra.instructions.ast.JumpInstruction.getInstruction()`.

Here is the call graph for this function:



### **6.66.3.3 int umbra.instructions.ast.JumpInstruction.getInd () [protected]**

This method parses the parameter of the current instruction.

This method retrieves the numerical value of the parameter of the instruction in [BytecodeLineController#getMy\\_line\\_text\(\)](#). This parameter is located after the mnemonic followed by #. (with no whitespace inbetween). The method assumes [BytecodeLineController#getMy\\_line\\_text\(\)](#) is correct.

## Returns:

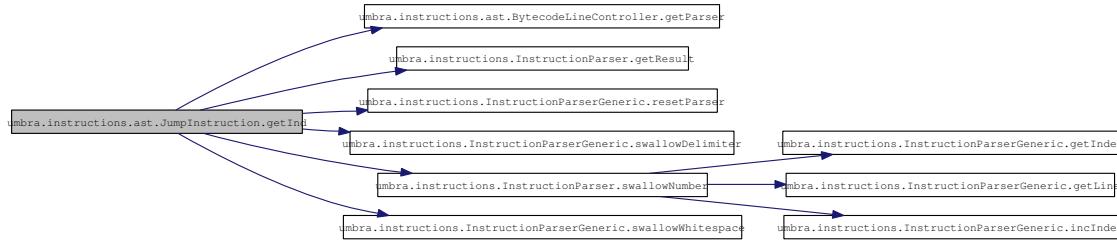
the value of the numerical parameter of the instruction

Reimplemented from [umbra.instructions.ast.MultiInstruction](#).

References `umbra.instructions.ast.BytecodeLineController.getParser()`, `umbra.instructions.InstructionParser.getResult()`, `umbra.instructions.InstructionParserGeneric.resetParser()`, `umbra.instructions.InstructionParserGeneric.swallowDelimiter()`, `umbra.instructions.InstructionParser.swallowNumber()`, and `umbra.instructions.InstructionParserGeneric.swallowWhitespace()`.

Referenced by `umbra.instructions.ast.JumpInstruction.setTarget()`.

Here is the call graph for this function:



#### 6.66.3.4 final Instruction umbra.instructions.ast.JumpInstruction.getInstruction ()

This method, based on the value of the mnemonic name, creates a new BCEL instruction object for a jump instruction, i.e.

2

- unconditional goto **instructions**,

- if `instructions` that compare references,
- if `instructions` that compare integers,
- if `instructions` that compare with null,
- if `instructions` that compare with 0,
- subroutine `instructions`.

This method also checks the syntactical correctness of the current instruction line.

#### Returns:

the freshly constructed BCCEL instruction or `null` in case the instruction is not a instruction from the current set and in case the instruction line is incorrect

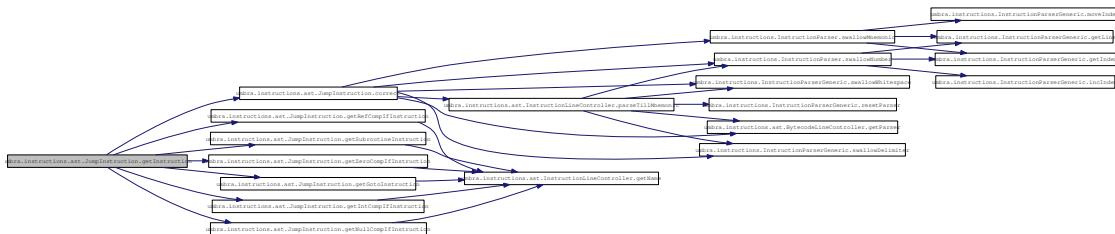
#### See also:

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References `umbra.instructions.ast.JumpInstruction.correct()`, `umbra.instructions.ast.JumpInstruction.getGotoInstruction()`, `umbra.instructions.ast.JumpInstruction.getIntCompIfInstruction()`, `umbra.instructions.ast.JumpInstruction.getNullCompIfInstruction()`, `umbra.instructions.ast.JumpInstruction.getRefCompIfInstruction()`, `umbra.instructions.ast.JumpInstruction.getSubroutineInstruction()`, `umbra.instructions.ast.JumpInstruction.getZeroCompIfInstruction()`.  
and `umbra.instructions.ast.JumpInstruction.getNullCompIfInstruction()`.

Here is the call graph for this function:



#### 6.66.3.5 Instruction `umbra.instructions.ast.JumpInstruction.getSubroutineInstruction (final InstructionHandle an_ih, final Instruction a_res) [private]`

This method creates the objects that represents a subroutine instruction.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array `instructions` are:

- `jsr`,
- `jsr_1`.

#### Parameters:

`an_ih` an instruction handle of the target instruction

*a\_res* a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or a\_res in case the current instruction is not in the current set

References umbra.instructions.ast.InstructionLineController.getName().

Referenced by umbra.instructions.ast.JumpInstruction.getInstruction().

Here is the call graph for this function:



#### 6.66.3.6 Instruction umbra.instructions.ast.JumpInstruction.getNullCompIfInstruction (final InstructionHandle *an\_ih*, final Instruction *a\_res*) [private]

This method creates the objects that represent if [instructions](#) that compare with null references.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter a\_res.

The array [instructions](#) are:

- ifnonnull,
- ifnull.

**Parameters:**

*an\_ih* an instruction handle of the target instruction

*a\_res* a helper value returned in case the current instruction is not in the current set

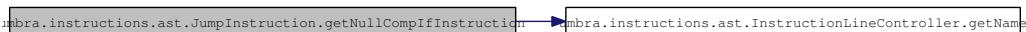
**Returns:**

the object that represents the current instruction or a\_res in case the current instruction is not in the current set

References umbra.instructions.ast.InstructionLineController.getName().

Referenced by umbra.instructions.ast.JumpInstruction.getInstruction().

Here is the call graph for this function:



#### 6.66.3.7 Instruction umbra.instructions.ast.JumpInstruction.getRefCompIfInstruction (final InstructionHandle *an\_ih*, final Instruction *a\_res*) [private]

This method creates the objects that represent if [instructions](#) that compare with references.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array `instructions` are:

- `if_acmpeq`,
- `if_acmpne`.

**Parameters:**

`an_ih` an instruction handle of the target instruction

`a_res` a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.JumpInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.66.3.8 Instruction `umbra.instructions.ast.JumpInstruction.getGotoInstruction` (final `InstructionHandle an_ih`, final `Instruction a_res`) [private]

This method creates the objects that represent goto `instructions`.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array `instructions` are:

- `goto`,
- `goto_w`.

**Parameters:**

`an_ih` an instruction handle of the target instruction

`a_res` a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.JumpInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.66.3.9 Instruction `umbra.instructions.ast.JumpInstruction.getIntCompIfInstruction (final InstructionHandle an_ih, final Instruction a_res) [private]`

This method creates the objects that represent if `instructions` to compare with integers.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array `instructions` are:

- `if_icmpeq`,
- `if_icmpge`,
- `if_icmpgt`,
- `if_icmple`,
- `if_icmplt`,
- `if_icmpne`.

#### Parameters:

`an_ih` an instruction handle of the target instruction

`a_res` a helper value returned in case the current instruction is not in the current set

#### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.JumpInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.66.3.10 Instruction `umbra.instructions.ast.JumpInstruction.getZeroCompIfInstruction (final InstructionHandle an_ih, final Instruction a_res) [private]`

This method creates the objects that represent if `instructions` that compare with integer 0.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array `instructions` are:

- ifeq,
- ifge,
- ifgt,
- ifle,
- iflt,
- ifne.

**Parameters:**

*an\_ih* an instruction handle of the target instruction

*a\_res* a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.JumpInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.66.3.11 final void umbra.instructions.ast.JumpInstruction.setTarget (final InstructionList an\_ins\_list, final Instruction an\_ins) throws UmbraException

Jump instruction requires an instruction number of its target as a parameter.

Note that the [BranchInstruction](#) has only one target.

**Parameters:**

*an\_ins\_list* an instruction list with the jump instruction  
*an\_ins* the jump instruction to set the target for

**Exceptions:**

[\*UmbraException\*](#) when the jump instruction has improper target

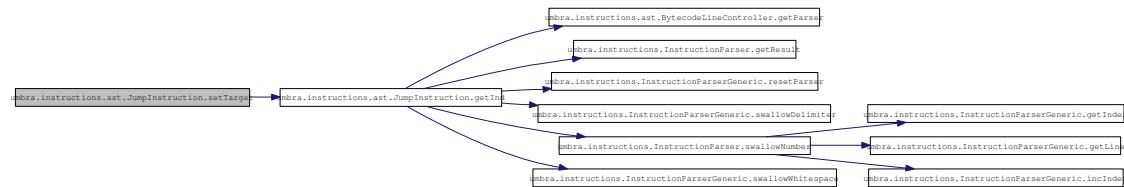
**See also:**

[umbra.instructions.ast.BytocodeLineController.setTarget\(](#) org.apache.bcel.generic.InstructionList,  
org.apache.bcel.generic Instruction)

Reimplemented from [umbra.instructions.ast.BytocodeLineController](#).

References `umbra.instructions.ast.JumpInstruction.getInd()`.

Here is the call graph for this function:



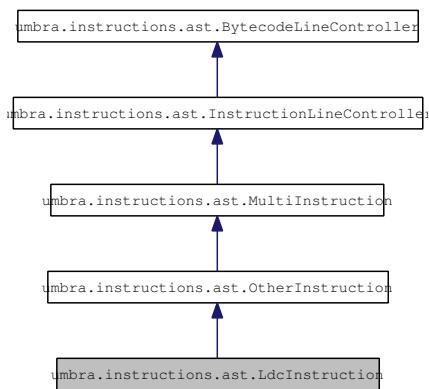
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[JumpInstruction.java](#)

## 6.67 umbra.instructions.ast.LdcInstruction Class Reference

This class is related to some subset of [instructions](#) depending on parameters.

Inheritance diagram for umbra.instructions.ast.LdcInstruction:



Collaboration diagram for umbra.instructions.ast.LdcInstruction:



### Public Member Functions

- [LdcInstruction](#) (final String a\_line\_text, final String a\_name)  
*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.*
- final Instruction [getInstruction](#) ()  
*This method, based on the value of the mnemonic name, creates a new BCEL instruction object for an LCD instruction, i.e.*
- final boolean [correct](#) ()  
*LDC instruction line is correct if it has one parameter that is a string enclosed by two " and another one that is a number in ().*

### Static Public Member Functions

- static String[] [getMnemonics](#) ()  
*This method returns the array of ldc [instructions](#) mnemonics.*

### Private Member Functions

- boolean [stringWithDelimiters](#) (final [InstructionParser](#) a\_parser)  
*This method tries to parse a string in ".*

### 6.67.1 Detailed Description

This class is related to some subset of [instructions](#) depending on parameters.

It redefines some crucial while handling with single instruction methods(correctness, getting handle). These instruction are dealing with long data.

**Author:**

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.67.2 Constructor & Destructor Documentation

#### 6.67.2.1 `umbra.instructions.ast.LdcInstruction.LdcInstruction (final String a_line_text, final String a_name)`

This creates an instance of an instruction named as `a_name` in a line the text of which is `a_line_text`.

Currently it just calls the constructor of the superclass.

**Parameters:**

`a_line_text` the line number of the instruction  
`a_name` the mnemonic name of the instruction

**See also:**

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.67.3 Member Function Documentation

#### 6.67.3.1 `static String [] umbra.instructions.ast.LdcInstruction.getMnemonics () [static]`

This method returns the array of ldc [instructions](#) mnemonics.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController](#).[getMnemonics](#)()

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

#### 6.67.3.2 `final Instruction umbra.instructions.ast.LdcInstruction.getInstruction ()`

This method, based on the value of the mnemonic name, creates a new BCEL instruction object for an LCD instruction, i.e.

:

- ldc,
  - ldc\_w,
  - ldc2\_w.

This method also checks the syntactical correctness of the current instruction line.

### Returns:

the freshly constructed BCEL instruction or `null` in case the instruction is not a instruction from the current set and in case the instruction line is incorrect

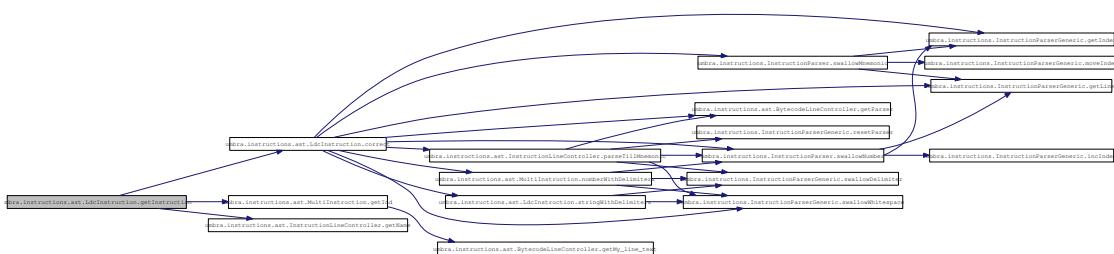
#### **See also:**

## BytecodeLineController.getInstruction()

Reimplemented from `umbra.instructions.ast.BytecodeLineController`.

References `umbra.instructions.ast.LdcInstruction.correct()`, `umbra.instructions.ast.MultiInstruction.getInd()`, and `umbra.instructions.ast.InstructionLineController.getName()`.

Here is the call graph for this function:



### 6.67.3.3 final boolean umbra.instructions.ast.LdcInstruction.correct()

LDC instruction line is correct if it has one parameter that is a string enclosed by two " and another one that is a number in () .

The precise format is: whitespace number : whitespace mnemonic whitespace " string " whitespace ( whitespace number whitespace ) whitespace lineend or whitespace number : whitespace mnemonic whitespace number whitespace ( whitespace number whitespace ) whitespace lineend

## Returns:

true when the syntax of the instruction line is correct

#### **See also:**

`InstructionLineController.correct()`

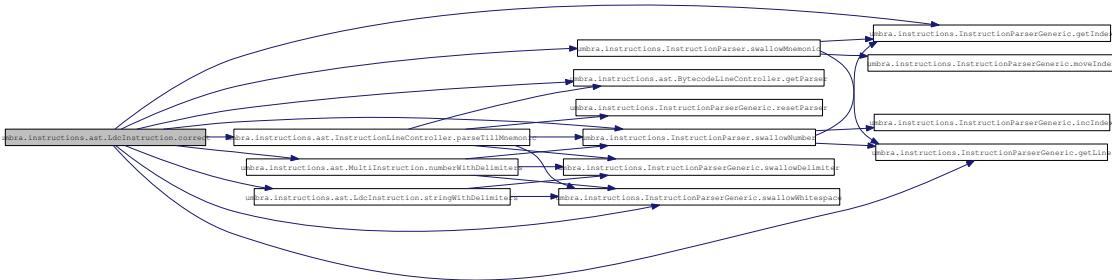
Reimplemented from `umbra.instructions.ast.InstructionLineController`.

References umbra.instructions.InstructionParserGeneric.getIndex(), umbra.instructions.InstructionParserGeneric.getLine(), umbra.instructions.ast.BytecodeLineController.getParser(),

umbra.instructions.ast.MultiInstruction.numberWithDelimiters(), umbra.instructions.ast.InstructionLineController.parseTillMnemonics(), umbra.instructions.ast.LdcInstruction.stringWithDelimiters(), umbra.instructions.InstructionParser.swallowMnemonic(), umbra.instructions.InstructionParser.swallowNumber(), and umbra.instructions.InstructionParserGeneric.swallowWhitespace()

Referenced by umbra.instructions.ast.LdcInstruction.getInstruction().

Here is the call graph for this function:



#### 6.67.3.4 boolean umbra.instructions.ast.LdcInstruction.stringWithDelimiters (final InstructionParser *a\_parser*) [private]

This method tries to parse a string in ".

The precise format is: " string "

##### Parameters:

*a\_parser* the parser which is to parse the string

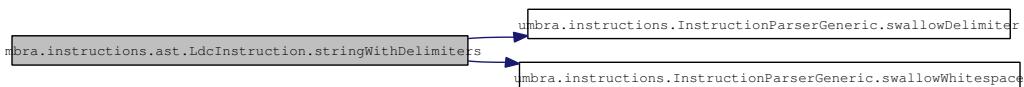
##### Returns:

true when the syntax of the string is correct

References umbra.instructions.InstructionParserGeneric.swallowDelimiter(), and umbra.instructions.InstructionParserGeneric.swallowWhitespace().

Referenced by umbra.instructions.ast.LdcInstruction.correct().

Here is the call graph for this function:



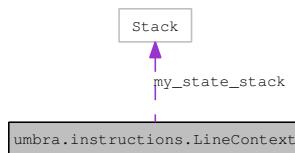
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[LdcInstruction.java](#)

## 6.68 umbra.instructions.LineContext Class Reference

The line parser on which the parsing of the byte code textual representation is based is context sensitive.

Collaboration diagram for umbra.instructions.LineContext:



### Public Member Functions

- [LineContext \(\)](#)

*The constructor initialises the internal state of the object so that it is in the internal state.*

- [void setInitial \(\)](#)

*This method sets the internal state of the object to the initial value.*

- [void setClassToBeRead \(\)](#)

*The method sets the internal state of the object to the state in which we are about to parse the class.*

- [boolean isInsideComment \(\)](#)

*Returns true when the object is in the state inside a comment.*

- [int getState \(\)](#)

*Returns the current state of the line context.*

- [void setInsideComment \(\)](#)

*Sets the current state to be the state inside a comment.*

- [void rememberState \(\)](#)

*It remembers the current state on the history stack state.*

- [void revertState \(\)](#)

*It restores from the history stack the previously remembered state.*

- [boolean isInsideAnnotation \(\)](#)

*Returns true when the object is in the state inside an annotation.*

- [void setInsideAnnotation \(final int a\\_pos\)](#)

*Sets the current state to be the state inside an annotation.*

- [void incMethodNo \(\)](#)

*This method advances by 1 the method number counter.*

- [int getMethodNo \(\)](#)

*This method returns the current method number.*

- void **setMethodNo** (final int a\_methodno)  
*This method initialises the internal method number to the given value.*
- int **getAnnotationEnd** ()  
*Returns the value of the remembered annotation end position.*
- void **setInsideMethod** ()  
*Sets the current state to be the state inside method.*
- void **setInvariantArea** ()  
*Sets the current state to be the state inside the invariant area.*
- boolean **isInInvariantArea** ()  
*Returns true when the object is in the state inside the invariant area.*
- boolean **isInsideMethod** ()  
*Returns true when the object is in the state inside the method.*

## Static Public Attributes

- static final int **STATE\_UNDEFINED** = 0  
*The context state which is used to mark an error situation.*

## Private Attributes

- int **my\_state**  
*The current state of the context.*
- Stack **my\_state\_stack**  
*The stack of states used to handle the parsing of comments.*
- int **my\_method**  
*The number of the currently parsed method.*
- int **my\_annotation\_end**  
*The last line in the annotation.*

## Static Private Attributes

- static final int **STATE\_INITIAL** = 1  
*The context state which is used at the begining of parsing.*
- static final int **STATE\_CLASS\_TO\_BE\_READ** = 2  
*The context state which is used in case we expect that the content of a class will be read.*

- static final int **STATE\_INSIDE\_COMMENT** = 3

*The context state which is used in case the parsing is inside of a multi-line comment.*

- static final int **STATE\_INSIDE\_ANNOTATION** = 4

*The context state which is used in case the parsing is inside of a BML annotation comment.*

- static final int **STATE\_INSIDE\_METHOD** = 5

*The context state which is used in case the parsing is inside of a method.*

- static final int **STATE\_INVARIANT\_AREA** = 6

*The context state which is used in case the parsing is inside of a method.*

### 6.68.1 Detailed Description

The line parser on which the parsing of the byte code textual representation is based is context sensitive.

In particular this representation can contain multi-line comments the contents of which should not be parsed. This class allows to keep track of all such issues. Currently it handles the cases when the parsing is:

- at the beginning of a text file,
- parses a class representation,
- parses a multi-line comment,
- parses a annotation comment.

Additionally, this keeps track of the parsed methods. This can be extended in the future to handle line number table etc.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.68.2 Constructor & Destructor Documentation

#### 6.68.2.1 umbra.instructions.LineContext.LineContext ()

The constructor initialises the internal state of the object so that it is in the internal state.

It also initialises the stack of states which must be reverted.

References `umbra.instructions.LineContext.my_method`, `umbra.instructions.LineContext.my_state_stack`, and `umbra.instructions.LineContext.setInitial()`.

Here is the call graph for this function:



### 6.68.3 Member Function Documentation

#### 6.68.3.1 void umbra.instructions.LineContext.setInitial ()

This method sets the internal state of the object to the initial value.

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.STATE_INITIAL`.

Referenced by `umbra.instructions.LineContext.LineContext()`.

#### 6.68.3.2 void umbra.instructions.LineContext.setClassToBeRead ()

The method sets the internal state of the object to the state in which we are about to parse the class.

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.STATE_CLASS_TO_BE_READ`.

#### 6.68.3.3 boolean umbra.instructions.LineContext.isInsideComment ()

Returns `true` when the object is in the state inside a comment.

**Returns:**

`true` when the object is in the state inside a comment `false` otherwise

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.STATE_INSIDE_COMMENT`.

Referenced by `umbra.instructions.BytecodeTextParser.extractCommentFromLine()`, and `umbra.instructions.Preparsing.getType()`.

#### 6.68.3.4 int umbra.instructions.LineContext.getState ()

Returns the current state of the line context.

**Returns:**

the current state of the line context

References `umbra.instructions.LineContext.my_state`.

#### 6.68.3.5 void umbra.instructions.LineContext.setInsideComment ()

Sets the current state to be the state inside a comment.

Additionally, this method remembers the current state so that it can be restored by `revertState()`.

References `umbra.instructions.LineContext.my_state`, `umbra.instructions.LineContext.rememberState()`, and `umbra.instructions.LineContext.STATE_INSIDE_COMMENT`.

Referenced by `umbra.instructions.Preparsing.getType()`.

Here is the call graph for this function:



#### 6.68.3.6 void umbra.instructions.LineContext.rememberState ()

It remembers the current state on the history stack state.

This functionality is needed to implement comments.

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.my_state_stack`.

Referenced by `umbra.instructions.LineContext.setInsideAnnotation()`, and `umbra.instructions.LineContext.setInsideComment()`.

#### 6.68.3.7 void umbra.instructions.LineContext.revertState ()

It restores from the history stack the previously remembered state.

This functionality is needed to implement comments.

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.my_state_stack`.

Referenced by `umbra.instructions.Preparsing.getType()`.

#### 6.68.3.8 boolean umbra.instructions.LineContext.isInsideAnnotation ()

Returns `true` when the object is in the state inside an annotation.

##### Returns:

`true` when the object is in the state inside an annotation `false` otherwise

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.STATE_INSIDE_ANNOTATION`.

Referenced by `umbra.instructions.Preparsing.getType()`.

#### 6.68.3.9 void umbra.instructions.LineContext.setInsideAnnotation (final int *a\_pos*)

Sets the current state to be the state inside an annotation.

Additionally, this method remembers the current state so that it can be restored by `revertState()`.

##### Parameters:

*a\_pos* the last `editor` line of the annotation to be parsed or -1

References `umbra.instructions.LineContext.my_annotation_end`, `umbra.instructions.LineContext.my_state`, `umbra.instructions.LineContext.rememberState()`, and `umbra.instructions.LineContext.STATE_INSIDE_ANNOTATION`.

Referenced by `umbra.instructions.BytocodeController.establishCurrentContext()`, and `umbra.instructions.Preparsing.getType()`.

Here is the call graph for this function:



**6.68.3.10 void umbra.instructions.LineContext.incMethodNo ()**

This method advances by 1 the method number counter.

Note that initially the method number is -1.

References umbra.instructions.LineContext.my\_method.

Referenced by umbra.instructions.InitParser.runParsing().

**6.68.3.11 int umbra.instructions.LineContext.getMethodNo ()**

This method returns the current method number.

**Returns:**

the current method number

References umbra.instructions.LineContext.my\_method.

Referenced by umbra.instructions.BytocodeTextParser.updateAnnotations().

**6.68.3.12 void umbra.instructions.LineContext.setMethodNo (final int *a\_methodno*)**

This method initialises the internal method number to the given value.

**Parameters:**

*a\_methodno* the method number to be set

References umbra.instructions.LineContext.my\_method.

Referenced by umbra.instructions.BytocodeController.establishCurrentContext().

**6.68.3.13 int umbra.instructions.LineContext.getAnnotationEnd ()**

Returns the value of the remembered annotation end position.

**Returns:**

the annotation end position

References umbra.instructions.LineContext.my\_annotation\_end.

**6.68.3.14 void umbra.instructions.LineContext.setInsideMethod ()**

Sets the current state to be the state inside method.

References umbra.instructions.LineContext.my\_state, and umbra.instructions.LineContext.STATE\_INSIDE\_METHOD.

Referenced by umbra.instructions.BytocodeController.establishCurrentContext().

**6.68.3.15 void umbra.instructions.LineContext.setInvariantArea ()**

Sets the current state to be the state inside the invariant area.

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.STATE_INVARIANT_AREA`.

Referenced by `umbra.instructions.BytecodeController.establishCurrentContext()`.

**6.68.3.16 boolean umbra.instructions.LineContext.isInInvariantArea ()**

Returns `true` when the object is in the state inside the invariant area.

**Returns:**

`true` when the object is in the state inside the invariant area, `false` otherwise

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.STATE_INVARIANT_AREA`.

**6.68.3.17 boolean umbra.instructions.LineContext.isInsideMethod ()**

Returns `true` when the object is in the state inside the method.

**Returns:**

`true` when the object is in the state inside a method, `false` otherwise

References `umbra.instructions.LineContext.my_state`, and `umbra.instructions.LineContext.STATE_INSIDE_METHOD`.

## 6.68.4 Member Data Documentation

**6.68.4.1 final int umbra.instructions.LineContext.STATE\_UNDEFINED = 0 [static]**

The context state which is used to mark an error situation.

**6.68.4.2 final int umbra.instructions.LineContext.STATE\_INITIAL = 1 [static, private]**

The context state which is used at the begining of parsing.

Referenced by `umbra.instructions.LineContext.setInitial()`.

**6.68.4.3 final int umbra.instructions.LineContext.STATE\_CLASS\_TO\_BE\_READ = 2 [static, private]**

The context state which is used in case we expect that the content of a class will be read.

Referenced by `umbra.instructions.LineContext.setClassToBeRead()`.

**6.68.4.4 final int umbra.instructions.LineContext.STATE\_INSIDE\_COMMENT = 3 [static, private]**

The context state which is used in case the parsing is inside of a multi-line comment.

Referenced by `umbra.instructions.LineContext.isInsideComment()`, and `umbra.instructions.LineContext.setInsideComment()`.

**6.68.4.5 final int umbra.instructions.LineContext.STATE\_INSIDE\_ANNOTATION = 4 [static, private]**

The context state which is used in case the parsing is inside of a BML annotation comment.

Referenced by `umbra.instructions.LineContext.isInsideAnnotation()`, and `umbra.instructions.LineContext.setInsideAnnotation()`.

**6.68.4.6 final int umbra.instructions.LineContext.STATE\_INSIDE\_METHOD = 5 [static, private]**

The context state which is used in case the parsing is inside of a method.

Referenced by `umbra.instructions.LineContext.isInsideMethod()`, and `umbra.instructions.LineContext.setInsideMethod()`.

**6.68.4.7 final int umbra.instructions.LineContext.STATE\_INVARIANT\_AREA = 6 [static, private]**

The context state which is used in case the parsing is inside of a method.

Referenced by `umbra.instructions.LineContext isInvariantArea()`, and `umbra.instructions.LineContext.setInvariantArea()`.

**6.68.4.8 int umbra.instructions.LineContext.my\_state [private]**

The current state of the context.

Referenced by `umbra.instructions.LineContext.getState()`, `umbra.instructions.LineContext.isInInvariantArea()`, `umbra.instructions.LineContext.isInsideAnnotation()`, `umbra.instructions.LineContext.isInsideComment()`, `umbra.instructions.LineContext.isInsideMethod()`, `umbra.instructions.LineContext.rememberState()`, `umbra.instructions.LineContext.revertState()`, `umbra.instructions.LineContext.setClassToBeRead()`, `umbra.instructions.LineContext.setInitial()`, `umbra.instructions.LineContext.setInsideAnnotation()`, `umbra.instructions.LineContext.setInsideComment()`, `umbra.instructions.LineContext.setInsideMethod()`, and `umbra.instructions.LineContext.setInvariantArea()`.

**6.68.4.9 Stack umbra.instructions.LineContext.my\_state\_stack [private]**

The stack of states used to handle the parsing of comments.

Referenced by `umbra.instructions.LineContext.LineContext()`, `umbra.instructions.LineContext.rememberState()`, and `umbra.instructions.LineContext.revertState()`.

**6.68.4.10 int umbra.instructions.LineContext.my\_method [private]**

The number of the currently parsed method.

It contains -1 in case the method number has not been set yet.

Referenced by umbra.instructions.LineContext.getMethodNo(),  
umbra.instructions.LineContext.incMethodNo(), umbra.instructions.LineContext(), and  
umbra.instructions.LineContext.setMethodNo().

**6.68.4.11 int umbra.instructions.LineContext.my\_annotation\_end [private]**

The last line in the annotation.

Referenced by umbra.instructions.LineContext.getAnnotationEnd(), and  
umbra.instructions.LineContext.setInsideAnnotation().

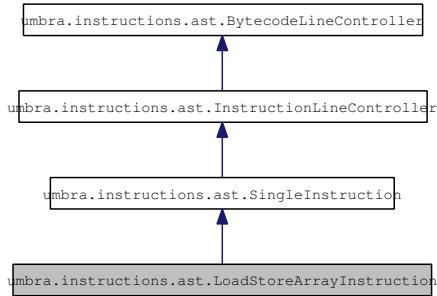
The documentation for this class was generated from the following file:

- source/umbra/instructions/[LineContext.java](#)

## 6.69 umbra.instructions.ast.LoadStoreArrayInstruction Class Reference

This class handles the creation and correctness for certain [instructions](#) with no parameters.

Inheritance diagram for `umbra.instructions.ast.LoadStoreArrayInstruction`:



Collaboration diagram for `umbra.instructions.ast.LoadStoreArrayInstruction`:



### Public Member Functions

- `LoadStoreArrayInstruction (final String a_line_text, final String a_name)`

*This creates an instance of an instruction named as a\_name with the line text a\_line.*

- `final Instruction getInstruction ()`

*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for an instruction with no parameters that loads or stores a for array entries.*

- `boolean correct ()`

*Simple instruction line is correct if it has no parameter.*

### Static Public Member Functions

- `static String[] getMnemonics ()`

*This method returns the array of load-store [instructions](#) mnemonics for arrays.*

### Private Member Functions

- `Instruction getArrayLoadStoreInstruction (finalInstruction a_res)`

*This method creates the objects that represent array load or store [instructions](#).*

- `Instruction getArrayShortLAInstruction (final Instruction a_res)`

*This method creates the objects that represent array load or store [instructions](#) for shorts.*

- Instruction [getArrayLongLSInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent array load or store [instructions](#) for longs.*

- Instruction [getArrayIntLSInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent array load or store [instructions](#) for ints.*

- Instruction [getArrayFloatLSInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent array load or store [instructions](#) for floats.*

- Instruction [getArrayDoubleLSInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent array load or store [instructions](#) for doubles.*

- Instruction [getArrayCharLSInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent array load or store [instructions](#) for chars.*

- Instruction [getArrayBoolLSInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent array load or store [instructions](#) for bytes.*

- Instruction [getArrayArrayLSInstruction](#) (final Instruction a\_res)

*This method creates the objects that represent array load or store [instructions](#) for arrays.*

## 6.69.1 Detailed Description

This class handles the creation and correctness for certain [instructions](#) with no parameters.

The [instructions](#) handled here are the [instructions](#) that load and store the contents of arrays.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

## 6.69.2 Constructor & Destructor Documentation

### 6.69.2.1 umbra.instructions.ast.LoadStoreArrayInstruction.LoadStoreArrayInstruction (final String a\_line\_text, final String a\_name)

This creates an instance of an instruction named as a\_name with the line text a\_line.

Currently it just calls the constructor of the superclass.

**Parameters:**

*a\_line\_text* the line number of the instruction

*a\_name* the mnemonic name of the instruction

**See also:**

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.69.3 Member Function Documentation

#### 6.69.3.1 static String [ ] umbra.instructions.ast.LoadStoreArrayInstruction.getMnemonics () [static]

This method returns the array of load-store [instructions](#) mnemonics for arrays.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController.getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

#### 6.69.3.2 Instruction um- bra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction (final Instruction *a\_res*) [private]

This method creates the objects that represent array load or store [instructions](#).

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The array load or store [instructions](#) are:

- [instructions](#) to l/s arrays,
- [instructions](#) to l/s bytes,
- [instructions](#) to l/s chars,
- [instructions](#) to l/s doubles,
- [instructions](#) to l/s floats,
- [instructions](#) to l/s ints,
- [instructions](#) to l/s longs,
- [instructions](#) to l/s shorts.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References	umbra.instructions.ast.LoadStoreArrayInstruction.getArrayArrayLSInstruction(), umbra.instructions.ast.LoadStoreArrayInstruction.getArrayBoolLSInstruction(), umbra.instructions.ast.LoadStoreArrayInstruction.getArrayCharLSInstruction(), umbra.instructions.ast.LoadStoreArrayInstruction.getArrayDoubleLSInstruction(), umbra.instructions.ast.LoadStoreArrayInstruction.getArrayFloatLSInstruction(), umbra.instructions.ast.LoadStoreArrayInstruction.getArrayIntLSInstruction(), umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLongLSInstruction(), umbra.instructions.ast.LoadStoreArrayInstruction.getArrayShortLAInstruction().	and	um-
------------	--	-----	-----

Referenced by `umbra.instructions.ast.LoadStoreArrayInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.69.3.3 Instruction `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayShortLAInstruction` (final Instruction `a_res`) [private]

This method creates the objects that represent array load or store [instructions](#) for shorts.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array load or store [instructions](#) for shorts are:

- `saload`,
- `sastore`.

#### Parameters:

`a_res` a helper value returned in case the current instruction is not in the current set

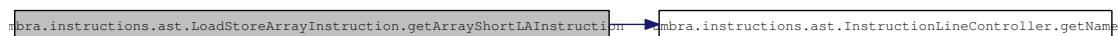
#### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction()`.

Here is the call graph for this function:



### 6.69.3.4 Instruction `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLongLSInstruction` (final Instruction `a_res`) [private]

This method creates the objects that represent array load or store [instructions](#) for longs.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array load or store [instructions](#) for loads are:

- laload,
- lastore.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

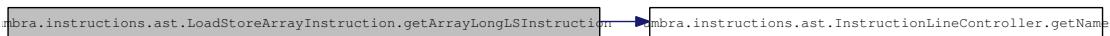
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction()`.

Here is the call graph for this function:



### 6.69.3.5 Instruction `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayIntLSInstruction` (final Instruction *a\_res*) [private]

This method creates the objects that represent array load or store [instructions](#) for ints.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The array load or store [instructions](#) for ints are:

- iaload,
- iastore.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

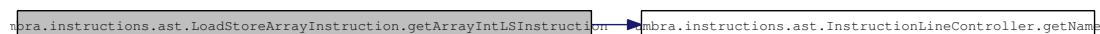
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction()`.

Here is the call graph for this function:



**6.69.3.6 Instruction um-****bra.instructions.ast.LoadStoreArrayInstruction.getArrayFloatLSInstruction (final  
Instruction *a\_res*) [private]**

This method creates the objects that represent array load or store [instructions](#) for floats.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The array load or store [instructions](#) for floats are:

- faload,
- fastore.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

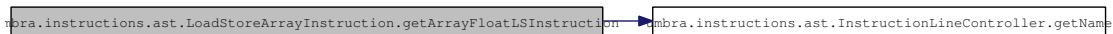
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References [umbra.instructions.ast.InstructionLineController.getName\(\)](#).

Referenced by [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction\(\)](#).

Here is the call graph for this function:

**6.69.3.7 Instruction um-****bra.instructions.ast.LoadStoreArrayInstruction.getArrayDoubleLSInstruction (final  
Instruction *a\_res*) [private]**

This method creates the objects that represent array load or store [instructions](#) for doubles.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The array load or store [instructions](#) for doubles are:

- daload,
- dastore.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

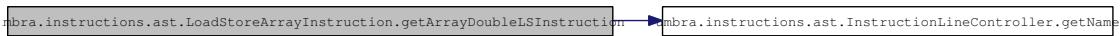
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction()`.

Here is the call graph for this function:



#### **6.69.3.8 Instruction um- bra.instructions.ast.LoadStoreArrayInstruction.getArrayCharLSInstruction (final Instruction a\_res) [private]**

This method creates the objects that represent array load or store `instructions` for chars.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array load or store `instructions` for chars are:

- caload,
- castore.

**Parameters:**

`a_res` a helper value returned in case the current instruction is not in the current set

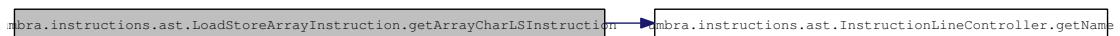
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction()`.

Here is the call graph for this function:



#### **6.69.3.9 Instruction um- bra.instructions.ast.LoadStoreArrayInstruction.getArrayBoolLSInstruction (final Instruction a\_res) [private]**

This method creates the objects that represent array load or store `instructions` for bytes.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array load or store `instructions` for bytes are:

- baload,

- bastore.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

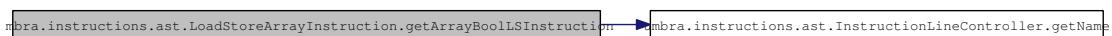
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction()`.

Here is the call graph for this function:



### 6.69.3.10 Instruction `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayArrayLSInstruction` (final Instruction *a\_res*) [private]

This method creates the objects that represent array load or store **instructions** for arrays.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The array load or store **instructions** for arrays are:

- aaload,
- aastore.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

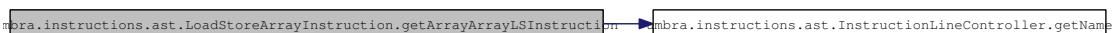
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction()`.

Here is the call graph for this function:



### 6.69.3.11 final Instruction umbra.instructions.ast.LoadStoreArrayInstruction.getInstruction ()

This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for an instruction with no parameters that loads or stores a for array entries.

This method also checks the syntactical correctness of the current instruction line.

#### Returns:

the freshly constructed BCEL instruction or `null` in case the instruction is not a instruction from the current set and in case the instruction line is incorrect

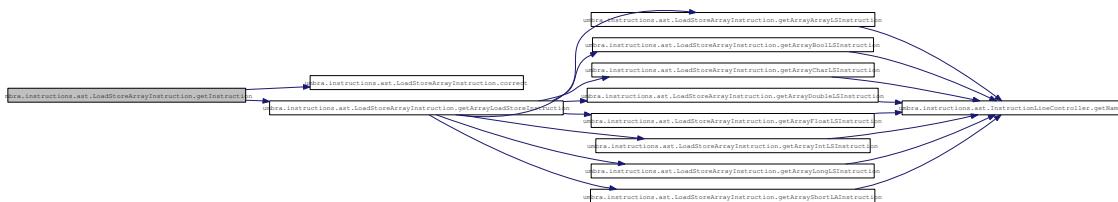
#### See also:

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

References      [umbra.instructions.ast.LoadStoreArrayInstruction.correct\(\)](#),      and      [umbra.instructions.ast.LoadStoreArrayInstruction.getArrayLoadStoreInstruction\(\)](#).

Here is the call graph for this function:



### 6.69.3.12 boolean umbra.instructions.ast.LoadStoreArrayInstruction.correct ()

Simple instruction line is correct if it has no parameter.

#### Returns:

`true` when the instruction mnemonic is the only text in the line of the instruction text

#### See also:

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

Referenced by [umbra.instructions.ast.LoadStoreArrayInstruction.getInstruction\(\)](#).

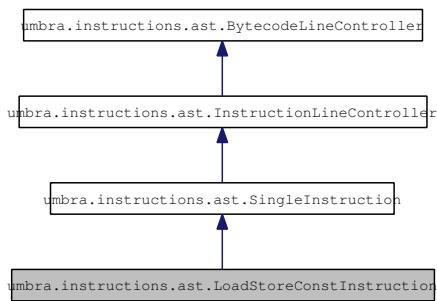
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[LoadStoreArrayInstruction.java](#)

## 6.70 umbra.instructions.ast.LoadStoreConstInstruction Class Reference

This class handles the creation and correctness for [instructions](#) with no parameters that perform loading and storing data on/from the operand stack.

Inheritance diagram for `umbra.instructions.ast.LoadStoreConstInstruction`:



Collaboration diagram for `umbra.instructions.ast.LoadStoreConstInstruction`:



### Public Member Functions

- `LoadStoreConstInstruction (final String a_line_text, final String a_name)`  
*This creates an instance of an instruction named as `a_name` with the line text `a_line`.*
- `final Instruction getInstruction ()`  
*This method, based on the value of the instruction line (from `InstructionLineController`), creates a new BCEL instruction object for an instruction with no parameters that loads or stores a for a constant value i.e.*
- `boolean correct ()`  
*Simple instruction line is correct if it has no parameter.*

### Static Public Member Functions

- `static String[] getMnemonics ()`  
*This method returns the array of load and store `instructions` mnemonics.*

### Private Member Functions

- `Instruction getConstLoadStoreInstruction (final Instruction a_res)`  
*This method creates the objects that represent `instructions` that load or store numbers and are parametrised by constants (e.g.*

- Instruction [getLLSInstruction](#) (finalInstruction a\_res, final int a\_num, finalString a\_name)  
*This method creates the objects that represent [instructions](#) that load or store long numbers and are parametrised by constants (e.g.*
- Instruction [getILSInstruction](#) (finalInstruction a\_res, final int a\_num, finalString a\_name)  
*This method creates the objects that represent [instructions](#) that load or store int numbers and are parametrised by constants (e.g.*
- Instruction [getFLSInstruction](#) (final Instruction a\_res, final int a\_num, final String a\_name)  
*This method creates the objects that represent [instructions](#) that load or store float numbers and are parametrised by constants (e.g.*
- Instruction [getDLSInstruction](#) (final Instruction a\_res, final int a\_num, final String a\_name)  
*This method creates the objects that represent [instructions](#) that load or store double numbers and are parametrised by constants (e.g.*
- Instruction [getALSInstruction](#) (final Instruction a\_res, final int a\_num, final String a\_name)  
*This method creates the objects that represent [instructions](#) that load or store references and are parametrised by constants (e.g.*

## Static Private Attributes

- static final int [MAX\\_LOAD\\_STORE\\_NUM](#) = 3  
*The constant that represents the maximal value of the constant parameter for [instructions](#) such as `iload-<n>`, see [getConstLoadStoreInstruction\(Instruction\)](#) for the full inventory.*

### 6.70.1 Detailed Description

This class handles the creation and correctness for [instructions](#) with no parameters that perform loading and storing data on/from the operand stack.

The [instructions](#) handled here are in the following form:

- `xload_<num>`,
- `xstore_<num>`.

where x is one of a, c, d, f l.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

## 6.70.2 Constructor & Destructor Documentation

### 6.70.2.1 umbra.instructions.ast.LoadStoreConstInstruction.LoadStoreConstInstruction (final String *a\_line\_text*, final String *a\_name*)

This creates an instance of an instruction named as *a\_name* with the line text *a\_line*.

Currently it just calls the constructor of the superclass.

#### Parameters:

*a\_line\_text* the line number of the instruction

*a\_name* the mnemonic name of the instruction

#### See also:

[InstructionLineControllerInstructionLineController\(String, String\)](#)

## 6.70.3 Member Function Documentation

### 6.70.3.1 static String [ ] umbra.instructions.ast.LoadStoreConstInstruction.getMnemonics () [static]

This method returns the array of load and store [instructions](#) mnemonics.

#### Returns:

the array of the handled mnemonics

#### See also:

[InstructionLineController.getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

### 6.70.3.2 Instruction um- bra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction (final Instruction *a\_res*) [private]

This method creates the objects that represent [instructions](#) that load or store numbers and are parametrised by constants (e.g.

`iload_0`). It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The load or store [instructions](#) that are parametrised by constants are:

- `aload_[0-3]`,
- `astore_[0-3]`,
- `dload_[0-3]`,
- `dstore_[0-3]`,
- `fload_[0-3]`,

- `fstore_[0-3]`,
- `iload_[0-3]`,
- `istore_[0-3]`,
- `lload_[0-3]`,
- `lstore_[0-3]`.

**Parameters:**

`a_res` a helper value returned in case the current instruction is not in the current set

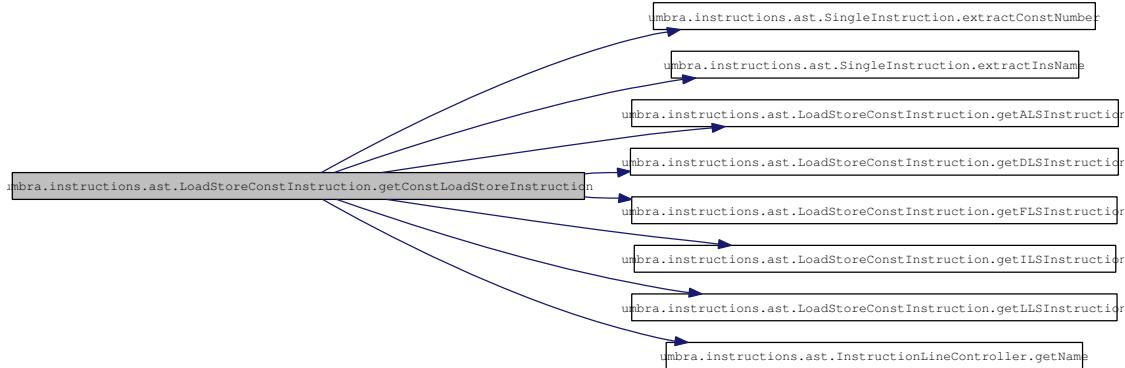
**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References `umbra.instructions.ast.SingleInstruction.extractConstNumber()`, `umbra.instructions.ast.SingleInstruction.extractInsName()`, `umbra.instructions.ast.LoadStoreConstInstruction.getALSIInstruction()`, `umbra.instructions.ast.LoadStoreConstInstruction.getDLSInstruction()`, `umbra.instructions.ast.LoadStoreConstInstruction.getFLSInstruction()`, `umbra.instructions.ast.LoadStoreConstInstruction.getILSInstruction()`, `umbra.instructions.ast.LoadStoreConstInstruction.getLLSInstruction()`, `umbra.instructions.ast.InstructionLineController.getName()`, and `umbra.instructions.ast.LoadStoreConstInstruction.MAX_LOAD_STORE_NUM`.

Referenced by `umbra.instructions.ast.LoadStoreConstInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.70.3.3 Instruction `umbra.instructions.ast.LoadStoreConstInstruction.getLLSInstruction (final Instruction a_res, final int a_num, final String a_name) [private]`

This method creates the objects that represent `instructions` that load or store long numbers and are parametrised by constants (e.g.

`lload_0`). It assumes all the checks are done in `getConstLoadStoreInstruction(Instruction)`. In case the name mentioned in `a_name` is of a different kind it returns the parameter `a_res`.

The load or store `instructions` for longs that are parametrised by constants are:

- lload\_[0-3],
- istore\_[0-3].

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set  
*a\_num* the number constant with which the instruction should be created  
*a\_name* the name of the instruction (with the number stripped, e.g. for lload\_0 it is lload)

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

Referenced by `umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction()`.

#### 6.70.3.4 Instruction `umbra.instructions.ast.LoadStoreConstInstruction.getILSInstruction (final Instruction a_res, final int a_num, final String a_name) [private]`

This method creates the objects that represent `instructions` that load or store int numbers and are parametrised by constants (e.g.

`iload_0`). It assumes all the checks are done in `getConstLoadStoreInstruction(Instruction)`. In case the name mentioned in *a\_name* is of a different kind it returns the parameter *a\_res*.

The load or store `instructions` for ints that are parametrised by constants are:

- iload\_[0-3],
- istore\_[0-3].

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set  
*a\_num* the number constant with which the instruction should be created  
*a\_name* the name of the instruction (with the number stripped, e.g. for iload\_0 it is iload)

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

Referenced by `umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction()`.

#### 6.70.3.5 Instruction `umbra.instructions.ast.LoadStoreConstInstruction.getFLSInstruction (final Instruction a_res, final int a_num, final String a_name) [private]`

This method creates the objects that represent `instructions` that load or store float numbers and are parametrised by constants (e.g.

`fload_0`). It assumes all the checks are done in `getConstLoadStoreInstruction(Instruction)`. In case the name mentioned in *a\_name* is of a different kind it returns the parameter *a\_res*.

The load or store `instructions` for floats that are parametrised by constants are:

- fload\_[0-3],
- fstore\_[0-3].

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set  
*a\_num* the number constant with which the instruction should be created  
*a\_name* the name of the instruction (with the number stripped, e.g. for fload\_0 it is fload)

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

Referenced by `umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction()`.

#### 6.70.3.6 Instruction `umbra.instructions.ast.LoadStoreConstInstruction.getDLSInstruction (final Instruction a_res, final int a_num, final String a_name) [private]`

This method creates the objects that represent `instructions` that load or store double numbers and are parametrised by constants (e.g.

`dload_0`). It assumes all the checks are done in `getConstLoadStoreInstruction(Instruction)`. In case the name mentioned in `a_name` is of a different kind it returns the parameter `a_res`.

The load or store `instructions` for doubles that are parametrised by constants are:

- dload\_[0-3],
- dstore\_[0-3].

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set  
*a\_num* the number constant with which the instruction should be created  
*a\_name* the name of the instruction (with the number stripped, e.g. for `dload_0` it is `dload`)

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

Referenced by `umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction()`.

#### 6.70.3.7 Instruction `umbra.instructions.ast.LoadStoreConstInstruction.getALSInstruction (final Instruction a_res, final int a_num, final String a_name) [private]`

This method creates the objects that represent `instructions` that load or store references and are parametrised by constants (e.g.

`aload_0`). It assumes all the checks are done in `getConstLoadStoreInstruction(Instruction)`. In case the name mentioned in `a_name` is of a different kind it returns the parameter `a_res`.

The load or store `instructions` for references that are parametrised by constants are:

- `aload_[0-3]`,
- `astore_[0-3]`.

**Parameters:**

`a_res` a helper value returned in case the current instruction is not in the current set  
`a_num` the number constant with which the instruction should be created  
`a_name` the name of the instruction (with the number stripped, e.g. for `lload_0` it is `lload`)

**Returns:**

the object that represents the current instruction or `res` in case the current instruction is not in the current set

Referenced by `umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction()`.

**6.70.3.8 final Instruction umbra.instructions.ast.LoadStoreConstInstruction.getInstruction ()**

This method, based on the value of the instruction line (from [InstructionLineController](#)), creates a new BCEL instruction object for an instruction with no parameters that loads or stores a for a constant value i.e.

- `xload_<number>`,
- `xstore_<number>`.

where `x` is one of `a, c, d, f l`.

This method also checks the syntactical correctness of the current instruction line.

**Returns:**

the freshly constructed BCEL instruction or `null` in case the instruction is not a instruction from the current set and in case the instruction line is incorrect

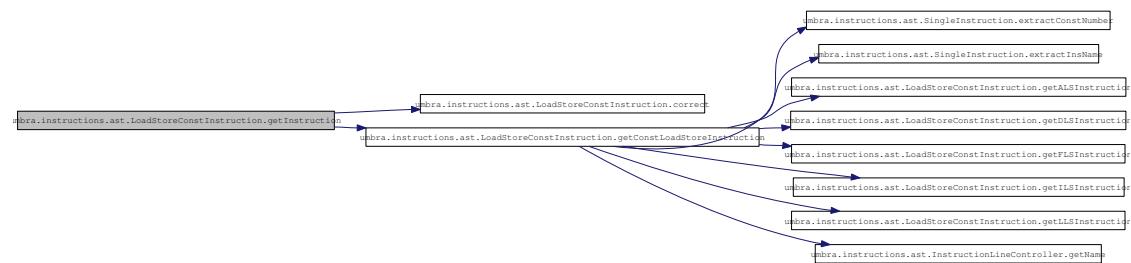
**See also:**

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

References [umbra.instructions.ast.LoadStoreConstInstruction.correct\(\)](#), and [umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction\(\)](#).

Here is the call graph for this function:



### 6.70.3.9 boolean **umbra.instructions.ast.LoadStoreConstInstruction.correct()**

Simple instruction line is correct if it has no parameter.

**Returns:**

true when the instruction mnemonic is the only text in the line of the instruction text

**See also:**

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.SingleInstruction](#).

Referenced by [umbra.instructions.ast.LoadStoreConstInstruction.getInstruction\(\)](#).

## 6.70.4 Member Data Documentation

### 6.70.4.1 final int **umbra.instructions.ast.LoadStoreConstInstruction.MAX\_LOAD\_STORE\_NUM = 3** [static, private]

The constant that represents the maximal value of the constant parameter for [instructions](#) such as `iload_-<n>`, see [getConstLoadStoreInstruction\(Instruction\)](#) for the full inventory.

Referenced by [umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction\(\)](#).

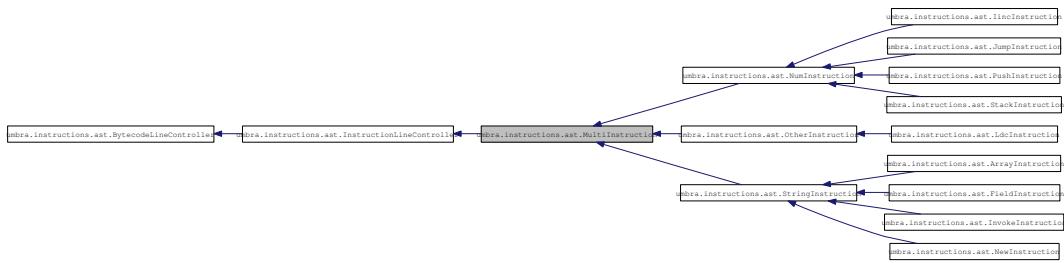
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[LoadStoreConstInstruction.java](#)

## 6.71 umbra.instructions.ast.MultiInstruction Class Reference

This is abstract class for all [instructions](#) with at least one parameter.

Inheritance diagram for `umbra.instructions.ast.MultiInstruction`:



Collaboration diagram for `umbra.instructions.ast.MultiInstruction`:



### Public Member Functions

- `MultiInstruction (final String a_line_text, final String a_name)`

*This creates an instance of an instruction named as a\_name at the line number a\_line\_text.*

### Static Public Member Functions

- static boolean `onlyDigitsInParen (finalString a_line_text)`

*This method checks if the last parenthesis in the given string contains only digits.*

- static int `getNumInParen (finalString a_line_text)`

*The method returns the number between the final parenthesis in the given string.*

### Protected Member Functions

- int `getInd ()`

*This method parses the parameter of the current instruction.*

- boolean `numberWithDelimiters (final InstructionParser a_parser)`

*This method tries to parse a number in () .*

#### 6.71.1 Detailed Description

This is abstract class for all [instructions](#) with at least one parameter.

**Author:**

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

## 6.71.2 Constructor & Destructor Documentation

### 6.71.2.1 **umbra.instructions.ast.MultiInstruction** (final String *a\_line\_text*, final String *a\_name*)

This creates an instance of an instruction named as *a\_name* at the line number *a\_line\_text*. Currently it just calls the constructor of the superclass.

**Parameters:**

*a\_line\_text* the line number of the instruction  
*a\_name* the mnemonic name of the instruction

**See also:**

InstructionLineControllerInstructionLineController(String, String)

## 6.71.3 Member Function Documentation

### 6.71.3.1 **static boolean umbra.instructions.ast.MultiInstruction.onlyDigitsInParen** (final String *a\_line\_text*) [static]

This method checks if the last parenthesis in the given string contains only digits.

**Parameters:**

*a\_line\_text* the string to check

**Returns:**

true when the last parenthesis contains only digits, false otherwise

### 6.71.3.2 **static int umbra.instructions.ast.MultiInstruction.getNumInParen** (final String *a\_line\_text*) [static]

The method returns the number between the final parenthesis in the given string.

It assumes that the string between the parenthesis indeed represents a number.

**Parameters:**

*a\_line\_text* a string to extract the number from

**Returns:**

the extracted number

### 6.71.3.3 int umbra.instructions.ast.MultiInstruction.getInd () [protected]

This method parses the parameter of the current instruction.

The default behaviour is that it parses the content of the final parenthesis in the instruction with a numeric argument. It checks if the argument is indeed the number and in that case it returns the number. In case the argument is not a number, the method returns 0. It also issues some logging information when the line has incorrect format.

#### Returns:

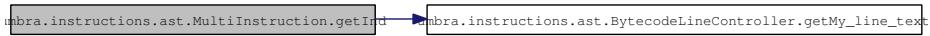
the parsed number or 0 in case the number cannot be parsed

Reimplemented in [umbra.instructions.ast.JumpInstruction](#), [umbra.instructions.ast.PushInstruction](#), and [umbra.instructions.ast.StackInstruction](#).

References [umbra.instructions.ast.BytecodeLineController.getMy\\_line\\_text\(\)](#), and [umbra.instructions.ast.BytecodeLineController.my\\_line\\_text](#).

Referenced by [umbra.instructions.ast.NewInstruction.getInstruction\(\)](#), [umbra.instructions.ast.LdcInstruction.getInstruction\(\)](#), [umbra.instructions.ast.InvokeInstruction.getInstruction\(\)](#), and [umbra.instructions.ast.FieldInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



### 6.71.3.4 boolean umbra.instructions.ast.MultiInstruction.numberWithDelimiters (final InstructionParser a\_parser) [protected]

This method tries to parse a number in ().

The precise format is: ( whitespace number whitespace )

#### Parameters:

*a\_parser* the parser which is to parse the number

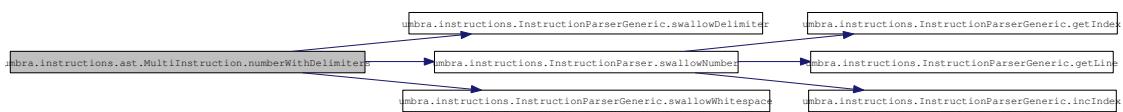
#### Returns:

true when the syntax of the number is correct

References [umbra.instructions.InstructionParserGeneric.swallowDelimiter\(\)](#), [umbra.instructions.InstructionParser.swallowNumber\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#).

Referenced by [umbra.instructions.ast.NewInstruction.correct\(\)](#), [umbra.instructions.ast.LdcInstruction.correct\(\)](#), [umbra.instructions.ast.InvokeInstruction.correct\(\)](#), and [umbra.instructions.ast.FieldInstruction.correct\(\)](#).

Here is the call graph for this function:



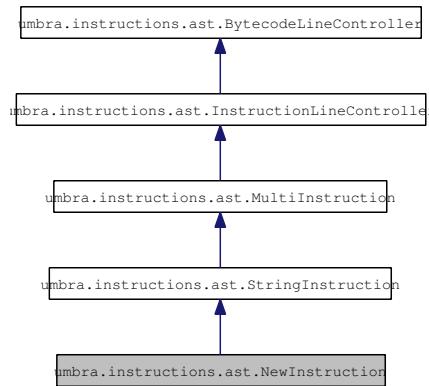
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[MultiInstruction.java](#)

## 6.72 umbra.instructions.ast.NewInstruction Class Reference

This class handles the creation and correctness for [instructions](#) to create objects, check their types, and cast them, namely::

Inheritance diagram for umbra.instructions.ast.NewInstruction:



Collaboration diagram for umbra.instructions.ast.NewInstruction:



### Public Member Functions

- [NewInstruction](#) (final String a\_line\_text, final String a\_name)

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.*

- final boolean [correct](#) ()

*New instruction line is correct if it has one parameter that is a class name in <> and another one that is a number in () .*

- final Instruction [getInstruction](#) ()

*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for a new-like instruction.*

### Static Public Member Functions

- static String[] [getMnemonics](#) ()

*This method returns the array of new [instructions](#) mnemonics.*

### Private Member Functions

- boolean [classnameWithDelimiters](#) (final [InstructionParser](#) a\_parser)

*This method tries to parse a class name in <>.*

### 6.72.1 Detailed Description

This class handles the creation and correctness for [instructions](#) to create objects, check their types, and cast them, namely::

- anewarray,
- checkcast,
- instanceof,
- new.

**Author:**

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.72.2 Constructor & Destructor Documentation

#### 6.72.2.1 **umbra.instructions.ast.NewInstruction (final String *a\_line\_text*, final String *a\_name*)**

This creates an instance of an instruction named as *a\_name* in a line the text of which is *a\_line\_text*. Currently it just calls the constructor of the superclass.

**Parameters:**

*a\_line\_text* the line number of the instruction  
*a\_name* the mnemonic name of the instruction

**See also:**

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.72.3 Member Function Documentation

#### 6.72.3.1 **static String [ ] umbra.instructions.ast.NewInstruction.getMnemonics () [static]**

This method returns the array of new [instructions](#) mnemonics.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController](#).[getMnemonics](#)()

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

#### 6.72.3.2 final boolean umbra.instructions.ast.NewInstruction.correct ()

New instruction line is correct if it has one parameter that is a class name in <> and another one that is a number in () .

The precise format is: whitespace number : whitespace mnemonic whitespace < whitespace classname whitespace > whitespace ( whitespace number whitespace ) whitespace lineend

## Returns:

true when the syntax of the instruction line is correct

#### **See also:**

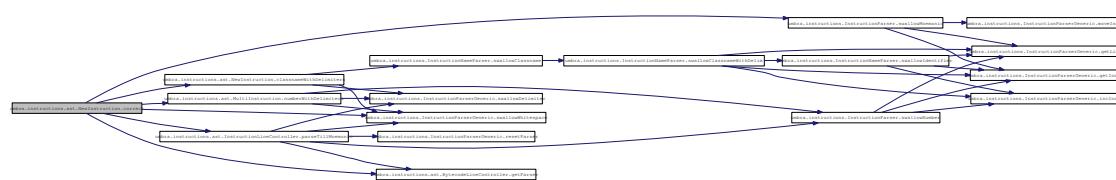
### InstructionLineController.correct()

Reimplemented from [umbra.instructions.astInstructionLineController](#).

References umbra.instructions.ast.NewInstruction.classnameWithDelimiters(), umbra.instructions.ast.BytecodeLineController.getParser(), umbra.instructions.ast.MultiInstruction.numberWithDelimiters(), umbra.instructions.ast.InstructionLineController.parseTillMnemonic(), umbra.instructions.InstructionParser.swallowMnemonic(), and umbra.instructions.InstructionParserGeneric.swallowWhitespace().

Referenced by `umbra.instructions.ast.NewInstruction.getInstruction()`.

Here is the call graph for this function:



**6.72.3.3 boolean umbra.instructions.ast.NewInstruction.classnameWithDelimiters (final InstructionParser a\_parser) [private]**

This method tries to parse a class name in <>.

The precise format is: < whitespace classname whitespace >

### Parameters:

*a parser* the parser which is to parse the class name

### Returns:

true when the syntax of the instruction line is correct

References [umbra.instructions.InstructionNameParser.swallowClassname\(\)](#),  
[umbra.instructions.InstructionParserGeneric.swallowDelimiter\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#)

Referenced by `umbra.instructions.ast.NewInstruction.correct()`

Here is the call graph for this function:



#### 6.72.3.4 final Instruction umbra.instructions.ast.NewInstruction.getInstruction ()

This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for a new-like instruction.

It computes the index parameter of the instruction before the instruction is constructed. The method can construct one of the [instructions](#):

- anewarray,
- checkcast,
- instanceof,
- new.

This method also checks the syntactical correctness of the current instruction line.

#### Returns:

the freshly constructed BCEL instruction or `null` in case the instruction is not a new-like instruction and in case the line is incorrect

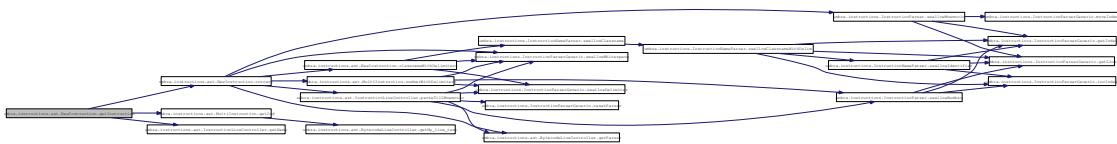
#### See also:

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References [umbra.instructions.ast.NewInstruction.correct\(\)](#), [umbra.instructions.ast.MultiInstruction.getInd\(\)](#), and [umbra.instructions.ast.InstructionLineController.getName\(\)](#).

Here is the call graph for this function:



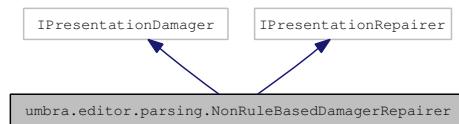
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[NewInstruction.java](#)

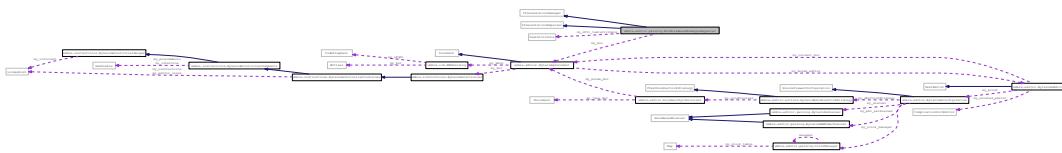
## 6.73 umbra.editor.parsing.NonRuleBasedDamagerRepairer Class Reference

This class is responsible for colouring these areas in a byte code [editor](#) window which are inside one-line areas.

Inheritance diagram for `umbra.editor.parsing.NonRuleBasedDamagerRepairer`:



Collaboration diagram for `umbra.editor.parsing.NonRuleBasedDamagerRepairer`:



### Public Member Functions

- `NonRuleBasedDamagerRepairer` (final `TextAttribute a_default_text_attribute`)  
*Constructor for [NonRuleBasedDamagerRepairer](#).*
- final void `setDocument` (final `IDocument a_doc`)  
*Associates the given document with the current damager-repairer.*
- final `IRegion getDamageRegion` (final `ITypedRegion a_partition`, final `DocumentEvent an_event`, final boolean `a_doc_partitioning_chngd`)  
*Returns the damage in the document's presentation caused by the current document change.*
- final void `createPresentation` (final `TextPresentation a_presentation`, final `ITypedRegion a_region`)  
*This method adds to a\_presentation a presentation style to be used to display a\_region.*

### Protected Member Functions

- final int `endOfLineOf` (final int `an_offset`) throws `UmbraLocationException`  
*Returns the end offset of the line that contains the specified offset or if the offset is inside a line delimiter, the end offset of the next line.*
- final void `addRange` (final `TextPresentation a_presentation`, final int `the_offset`, final int `the_length`, final `TextAttribute an_attr`)  
*Adds style information to the given text presentation.*

## Private Attributes

- `BytecodeDocument my_doc`

*The document this object works on.*

- `TextAttribute my_dflt_textattribute`

*The default text attribute used for the colouring of all the areas governed by the current object.*

### 6.73.1 Detailed Description

This class is responsible for colouring these areas in a byte code `editor` window which are inside one-line areas.

#### Author:

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

Wojciech Was ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

#### Version:

a-01

### 6.73.2 Constructor & Destructor Documentation

#### 6.73.2.1 `umbra.editor.parsing.NonRuleBasedDamagerRepairer.NonRuleBasedDamagerRepairer (final TextAttribute a_default_text_attribute)`

Constructor for `NonRuleBasedDamagerRepairer`.

It only caches the default text attribute.

#### Parameters:

`a_default_text_attribute` the default text attribute to be used by the current object

References `umbra.editor.parsing.NonRuleBasedDamagerRepairer.my_dflt_textattribute`.

### 6.73.3 Member Function Documentation

#### 6.73.3.1 `final void umbra.editor.parsing.NonRuleBasedDamagerRepairer.setDocument (final IDocument a_doc)`

Associates the given document with the current damager-repairer.

#### Parameters:

`a_doc` a document to associate with the current damager-repairer.

#### See also:

`IPresentationRepairer.setDocument(IDocument)`

References `umbra.editor.parsing.NonRuleBasedDamagerRepairer.my_doc`.

**6.73.3.2 final int umbra.editor.parsing.NonRuleBasedDamagerRepairer.endOfLineOf (final int  
*an\_offset*) throws UmbraLocationException [protected]**

Returns the end offset of the line that contains the specified offset or if the offset is inside a line delimiter, the end offset of the next line.

**Parameters:**

*an\_offset* the offset whose line end offset must be computed

**Returns:**

the line end offset for the given offset

**Exceptions:**

*UmbraLocationException* if the offset is invalid in the current document

References umbra.editor.parsing.NonRuleBasedDamagerRepairer.my\_doc.

Referenced by umbra.editor.parsing.NonRuleBasedDamagerRepairer.getDamageRegion().

**6.73.3.3 final IRegion umbra.editor.parsing.NonRuleBasedDamagerRepairer.getDamageRegion  
(final ITypedRegion *a\_partition*, final DocumentEvent *an\_event*, final boolean  
*a\_doc\_partitioning\_chngd*)**

Returns the damage in the document's presentation caused by the current document change.

In case the partitioning changed *a\_partition* is returned. In case the partitioning is unchanged the region is calculated which starts with the beginning of the line in which the modification started and ends with the end of the last line in which the modification occurred. The region is always included in the given damaged region so we have to check for cases in which the region starts/end in the middle of a line.

**Parameters:**

*a\_partition* a region which is damaged

*an\_event* the event which changes the document

*a\_doc\_partitioning\_chngd* true when the change changed document's partitioning

**Returns:**

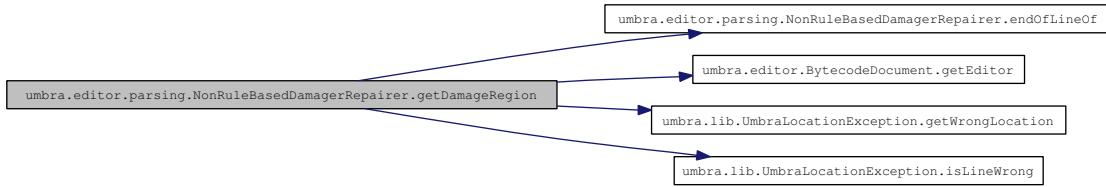
a new partition

**See also:**

IPresentationDamager.getDamageRegion(ITypedRegion, DocumentEvent, boolean)

References umbra.editor.parsing.NonRuleBasedDamagerRepairer.endOfLineOf(), umbra.editor.BytocodeDocument.getEditor(), umbra.lib.UmbraLocationException.getWrongLocation(), umbra.lib.UmbraLocationException.isLineWrong(), and umbra.editor.parsing.NonRuleBasedDamagerRepairer.my\_doc.

Here is the call graph for this function:



#### 6.73.3.4 final void umbra.editor.parsing.NonRuleBasedDamagerRepairer.createPresentation (final TextPresentation *a\_presentation*, final ITypedRegion *a\_region*)

This method adds to *a\_presentation* a presentation style to be used to display *a\_region*.

The presentation style is defined with the use of the default attribute.

**Parameters:**

- a\_presentation* the text presentation to be filled by this repairer
- a\_region* the damage to be repaired

**See also:**

IPresentationRepairer.createPresentation(TextPresentation, ITypedRegion)

References        `umbra.editor.parsing.NonRuleBasedDamagerRepairer.addRange()`,        and  
`umbra.editor.parsing.NonRuleBasedDamagerRepairer.my_dflt_textattribute`.

Here is the call graph for this function:



#### 6.73.3.5 final void umbra.editor.parsing.NonRuleBasedDamagerRepairer.addRange (final TextPresentation *a\_presentation*, final int *the\_offset*, final int *the\_length*, final TextAttribute *an\_attr*) [protected]

Adds style information to the given text presentation.

**Parameters:**

- a\_presentation* the text presentation to be extended
- the\_offset* the offset of the range to be styled
- the\_length* the length of the range to be styled
- an\_attr* the attribute describing the style of the range to be styled

Referenced by `umbra.editor.parsing.NonRuleBasedDamagerRepairer.createPresentation()`.

## 6.73.4 Member Data Documentation

### 6.73.4.1 BytecodeDocument umbra.editor.parsing.NonRuleBasedDamagerRepairer.my\_doc [private]

The document this object works on.

Referenced by umbra.editor.parsing.NonRuleBasedDamagerRepairer.endOfLineOf(), umbra.editor.parsing.NonRuleBasedDamagerRepairer.getDamageRegion(), and umbra.editor.parsing.NonRuleBasedDamagerRepairer.setDocument().

### 6.73.4.2 TextAttribute umbra.editor.parsing.NonRuleBasedDamagerRepairer.my\_dflt\_textattribute [private]

The default text attribute used for the colouring of all the areas governed by the current object.

Referenced by umbra.editor.parsing.NonRuleBasedDamagerRepairer.createPresentation(), and umbra.editor.parsing.NonRuleBasedDamagerRepairer.NonRuleBasedDamagerRepairer().

The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/[NonRuleBasedDamagerRepairer.java](#)

## 6.74 umbra.instructions.ast.NumInstruction Class Reference

This is abstract class for all [instructions](#) with a number as a parameter.

Inheritance diagram for `umbra.instructions.ast.NumInstruction`:



Collaboration diagram for `umbra.instructions.ast.NumInstruction`:



### Public Member Functions

- `NumInstruction (final String a_line_text, final String a_name)`

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.*

### Protected Member Functions

- int `checkInstructionWithNumber (final String a_line, final String an_instr, final char a_char_label)`  
*The method checks if the instruction an\_instr occurs correctly formatted in the line a\_line.*

### Private Member Functions

- int `checkNoParameters (final InstructionParser a_parser)`

*This method counts the number of parameters in the instruction parsed by a\_parser.*

### Static Private Attributes

- static final int `PARAMS_ONE = 1`  
*The constant to indicate one that the instruction has one parameter.*
- static final int `PARAMS_TWO = 2`  
*The constant to indicate one that the instruction has two parameters.*

#### 6.74.1 Detailed Description

This is abstract class for all [instructions](#) with a number as a parameter.

**Author:**

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

## 6.74.2 Constructor & Destructor Documentation

### 6.74.2.1 umbra.instructions.ast.NumInstruction NumInstruction (final String *a\_line\_text*, final String *a\_name*)

This creates an instance of an instruction named as *a\_name* in a line the text of which is *a\_line\_text*. Currently it just calls the constructor of the superclass.

**Parameters:**

*a\_line\_text* the line number of the instruction  
*a\_name* the mnemonic name of the instruction

**See also:**

InstructionLineControllerInstructionLineController(String, String)

## 6.74.3 Member Function Documentation

### 6.74.3.1 int umbra.instructions.ast.NumInstruction.checkInstructionWithNumber (final String *a\_line*, final String *an\_instr*, final char *a\_char\_label*) [protected]

The method checks if the instruction *an\_instr* occurs correctly formatted in the line *a\_line*.

The line is correct when it has the format whitespace number : whitespace mnemonic whitespace *a\_char\_label* whitespace number whitespace [number] whitespace lineend

**Parameters:**

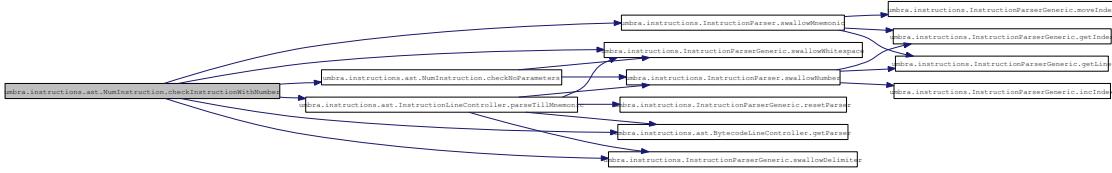
*a\_line* a bytecode line with all the whitespace stripped  
*an\_instr* an instruction text to be checked  
*a\_char\_label* the character which delimits the number

**Returns:**

-1 when we know that the syntax is wrong, 1 when we know the syntax is OK, 0 when we do not know, but we must search further

References      [umbra.instructions.ast.NumInstruction.checkNoParameters\(\)](#), [umbra.instructions.ast.BytecodeLineController.getParser\(\)](#), [umbra.instructions.ast.InstructionLineController.parseTillMnemonic\(\)](#), [umbra.instructions.InstructionParserGeneric.swallowDelimiter\(\)](#), [umbra.instructions.InstructionParser.swallowMnemonic\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#).

Here is the call graph for this function:



**6.74.3.2 int umbra.instructions.ast.NumInstruction.checkNoParameters (final InstructionParser a\_parser) [private]**

This method counts the number of parameters in the instruction parsed by `a_parser`.

We assume the index of the parser is situated so that the first number is about to be read (with no whitespace before that). We try then to read the first number and in case there is still something in the line the second number.

## Parameters:

*a\_parser* the parser which parses the analysed string

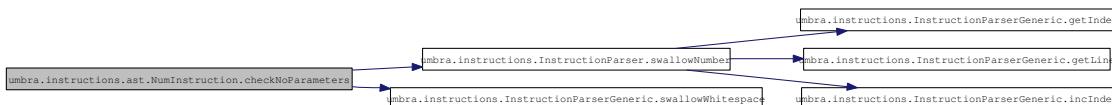
## Returns:

1, 2 mean one and two parameters resp., in other cases -1 is returned

References `umbra.instructions.ast.NumInstruction.PARAMS_ONE`, `umbra.instructions.ast.NumInstruction.PARAMS_TWO`, `umbra.instructions.InstructionParser.swallowNumber()`, and `umbra.instructions.InstructionParserGeneric.swallowWhitespace()`.

Referenced by `umbra.instructions.ast.NumInstruction.checkInstructionWithNumber()`.

Here is the call graph for this function:



#### **6.74.4 Member Data Documentation**

**6.74.4.1 final int umbra.instructions.ast.NumInstruction.PARAMS\_ONE = 1 [static, private]**

The constant to indicate one that the instruction has one parameter.

Referenced by `umbra.instructions.ast.NumInstruction.checkNoParameters()`.

**6.74.4.2 final int umbra.instructions.ast.NumInstruction.PARAMS\_TWO = 2** [static, private]

The constant to indicate one that the instruction has two parameters.

Referenced by `umbra.instructions.ast.NumInstruction.checkNoParameters()`.

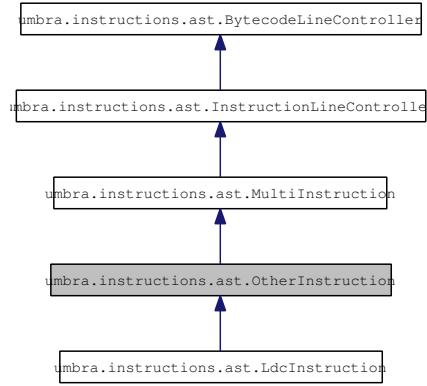
The documentation for this class was generated from the following file:

- `source/umbra/instructions/ast/NumInstruction.java`

## 6.75 umbra.instructions.ast.OtherInstruction Class Reference

This is abstract class for all [instructions](#) which are correct with number parameter as well as with a string one (in "").

Inheritance diagram for `umbra.instructions.ast.OtherInstruction`:



Collaboration diagram for `umbra.instructions.ast.OtherInstruction`:



### Public Member Functions

- [OtherInstruction](#) (final String a\_line\_text, final String a\_name)

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.*

#### 6.75.1 Detailed Description

This is abstract class for all [instructions](#) which are correct with number parameter as well as with a string one (in "").

##### Author:

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

##### Version:

a-01

#### 6.75.2 Constructor & Destructor Documentation

##### 6.75.2.1 `umbra.instructions.ast.OtherInstruction` (final String a\_line\_text, final String a\_name)

This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.

Currently it just calls the constructor of the superclass.

**Parameters:**

- *a\_line\_text* the line number of the instruction
- *a\_name* the mnemonic name of the instruction

**See also:**

`InstructionLineController`.`InstructionLineController`(String, String)

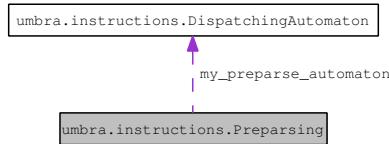
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[OtherInstruction.java](#)

## 6.76 umbra.instructions.Preparsing Class Reference

This class handles the preparing of document lines.

Collaboration diagram for umbra.instructions.Preparsing:



### Static Public Member Functions

- static [BytecodeLineController getType](#) (final String a\_line, final [LineContext](#) a\_context)  
*Chooses one of line types that matches the given line contents.*
- static [DispatchingAutomaton getAutomaton](#) ()  
*This method returns the automaton which handles the preparing of lines and creates appropriate line controllers.*

### Private Member Functions

- [Preparsing \(\)](#)  
*Private constructor added to prevent the creation of objects of this type.*

### Static Private Member Functions

- static void [addSimpleForArray](#) (final String[ ] the\_paths, final Class a\_class)  
*This method adds to the initial state of the preparing automaton the all the paths which are described by characters from the given array.*
- static void [addWhitespaceLoop](#) (final [DispatchingAutomaton](#) a\_state)  
*This method adds a whitespace loop to the given state of an automaton.*
- static void [addAllMnemonics](#) (final [DispatchingAutomaton](#) a\_node)  
*This method adds all the paths to recognise byte code mnemonics to the given node of an automaton.*

### Static Private Attributes

- static [DispatchingAutomaton my\\_preparse\\_automaton](#)  
*The automaton to pre-parse the lines of the byte code document.*

### 6.76.1 Detailed Description

This class handles the preparing of document lines.

It creates an automaton which recognises the particular line kind and creates the line handler. This automaton is used to obtain the line handler for the given string. Additionally, the process of getting of a line handler is controlled by a document context. In particular, the context recognises situation when the parsing is inside of a multi-line comment or a BML annotation.

**Author:**

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

### 6.76.2 Constructor & Destructor Documentation

#### 6.76.2.1 umbra.instructions.Preparsing () [private]

Private constructor added to prevent the creation of objects of this type.

### 6.76.3 Member Function Documentation

#### 6.76.3.1 static BytecodeLineController umbra.instructions.Preparsing.getType (final String *a\_line*, final LineContext *a\_context*) [static]

Chooses one of line types that matches the given line contents.

This method does a quick pre-parsing of the line content and based on that chooses which particular line controller should be used for the given line. It also uses the context information to return controllers in case the analysis is inside a comment or a BML annotation.

**Parameters:**

*a\_line* the string contents of inserted or modified line

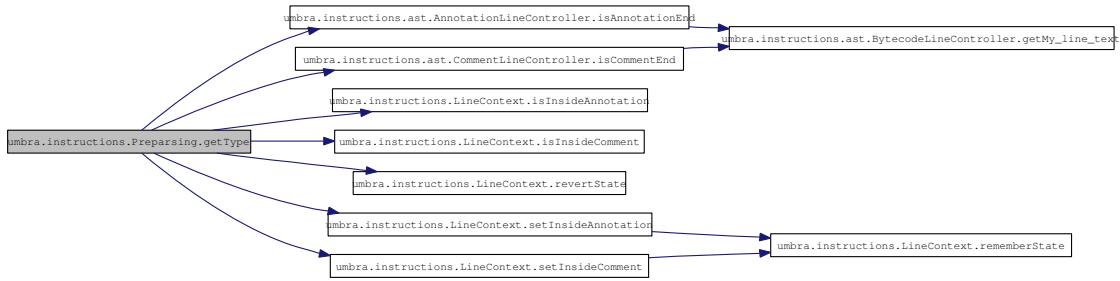
*a\_context* information on the previous lines

**Returns:**

instance of subclass of a line controller that contents of the given line satisfies classification conditions (unknown if it does not for all)

References      `umbra.instructions.ast.AnnotationLineController.isAnnotationEnd()`, `umbra.instructions.ast.CommentLineController.isCommentEnd()`, `umbra.instructions.LineContext.isInsideAnnotation()`, `umbra.instructions.LineContext.isInsideComment()`, `umbra.instructions.LineContext.revertState()`, `umbra.instructions.LineContext.setInsideAnnotation()`, and `umbra.instructions.LineContext.setInsideComment()`.

Here is the call graph for this function:



### 6.76.3.2 static DispatchingAutomaton `umbra.instructions.Preparsing.getAutomaton ()` [static]

This method returns the automaton which handles the preparing of lines and creates appropriate line controllers.

In case the automaton has not been created yet, the method creates it.

The automaton has the following major states:

- INITIAL - where all the processing starts
- DIGIT - where the digits of the byte code instruction number are recognised,
- COLON - after the colon of the byte code instruction is swallowed,
- many MNEMONIC states - to recognise mnemonics,
- many THROWS states - to recognise throws lines,
- many HEADER states - to recognise throws lines,
- COMMENT - to recognise multi-line comment start,
- ANNOT - to recognise BML annotation start.

The INITIAL state contains a loop over whitespace characters and outgoing edges (paths) to THROWS, HEADER, COMMENT, ANNOT and DIGIT states. The DIGIT state contains a loop over digits and an outgoing edge to the COLON state. The COLON state contains a loop over whitespace characters and outgoing edges to MNEMONIC states (paths to be precise).

Note that this automaton is slightly inefficient as MNEMONIC, THROWS etc. states could be made a single one.

#### Returns:

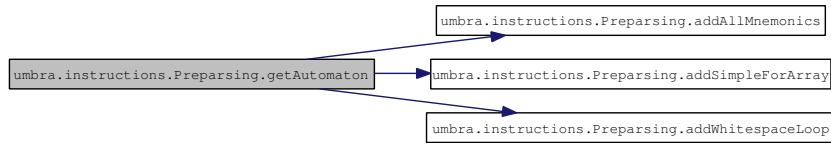
the automaton to handle preparsing of lines

#### See also:

[DispatchingAutomaton](#) for a description of the way the automaton works

References `umbra.instructions.Preparsing.addAllMnemonics()`, `umbra.instructions.Preparsing.addSimpleForArray()`, `umbra.instructions.Preparsing.addWhitespaceLoop()`, and `umbra.instructions.Preparsing.my_parse_automaton`.

Here is the call graph for this function:



### 6.76.3.3 static void umbra.instructions.Preparsing.addSimpleForArray (final String[ ] *the\_paths*, final Class *a\_class*) [static, private]

This method adds to the initial state of the preparsing automaton the all the paths which are described by characters from the given array.

The method associates the given class as the class the objects of which are created when the end of the path is reached in the automaton.

#### Parameters:

*the\_paths* the description of paths to be added

*a\_class* the class the objects of which should be created when the parsing reaches the terminal nodes created by this method

References `umbra.instructions.Preparsing.my_parse_automaton`.

Referenced by `umbra.instructions.Preparsing.getAutomaton()`.

### 6.76.3.4 static void umbra.instructions.Preparsing.addWhitespaceLoop (final DispatchingAutomaton *a\_state*) [static, private]

This method adds a whitespace loop to the given state of an automaton.

#### Parameters:

*a\_state* the state of the automaton

Referenced by `umbra.instructions.Preparsing.getAutomaton()`.

### 6.76.3.5 static void umbra.instructions.Preparsing.addAllMnemonics (final DispatchingAutomaton *a\_node*) [static, private]

This method adds all the paths to recognise byte code mnemonics to the given node of an automaton.

#### Parameters:

*a\_node* the node of the automaton to add the paths to

Referenced by `umbra.instructions.Preparsing.getAutomaton()`.

## 6.76.4 Member Data Documentation

### 6.76.4.1 DispatchingAutomaton umbra.instructions.Preparsing.my\_parse\_automaton [static, private]

The automaton to pre-parse the lines of the byte code document.

Referenced by umbra.instructions.Preparsing.addSimpleForArray(), and umbra.instructions.Preparsing.getAutomaton().

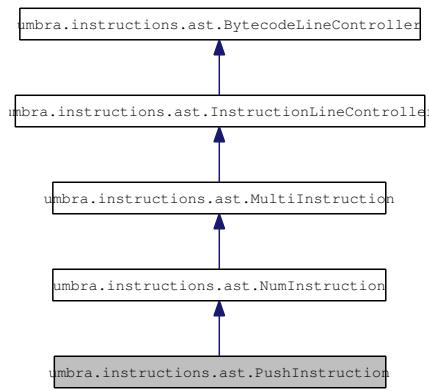
The documentation for this class was generated from the following file:

- source/umbra/instructions/[Preparsing.java](#)

## 6.77 umbra.instructions.ast.PushInstruction Class Reference

This class handles the creation and correctness for push [instructions](#) i.e.

Inheritance diagram for umbra.instructions.ast.PushInstruction:



Collaboration diagram for umbra.instructions.ast.PushInstruction:



### Public Member Functions

- [PushInstruction](#) (final String a\_line\_text, final String a\_name)

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line.*

- final boolean [correct](#) ()

*Push instruction line is correct if it has one simple number parameter.*

- final Instruction [getInstruction](#) ()

*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for a push instruction.*

### Static Public Member Functions

- static String[] [getMnemonics](#) ()

*This method returns the array of push [instructions](#) mnemonics.*

### Protected Member Functions

- int [getInd](#) ()

*This method parses the parameter of the current instruction.*

### 6.77.1 Detailed Description

This class handles the creation and correctness for push [instructions](#) i.e.

:

- bipush,
- sipush.

**Author:**

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.77.2 Constructor & Destructor Documentation

#### 6.77.2.1 [umbra.instructions.ast.PushInstruction.PushInstruction \(final String a\\_line\\_text, final String a\\_name\)](#)

This creates an instance of an instruction named as `a_name` in a line the text of which is `a_line`.

Currently it just calls the constructor of the superclass.

**Parameters:**

- `a_line_text` the line number of the instruction  
`a_name` the mnemonic name of the instruction

**See also:**

[InstructionLineController.InstructionLineController\(String, String\)](#)

### 6.77.3 Member Function Documentation

#### 6.77.3.1 [static String \[ \] umbra.instructions.ast.PushInstruction.getMnemonics \(\) \[static\]](#)

This method returns the array of push [instructions](#) mnemonics.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController.getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

### 6.77.3.2 final boolean umbra.instructions.ast.PushInstruction.correct ()

Push instruction line is correct if it has one simple number parameter.

The exact definition of this kind of a line is as follows: whitespace number : whitespace mnemonic whitespace number whitespace lineend

#### Returns:

true when the syntax of the instruction line is correct

#### See also:

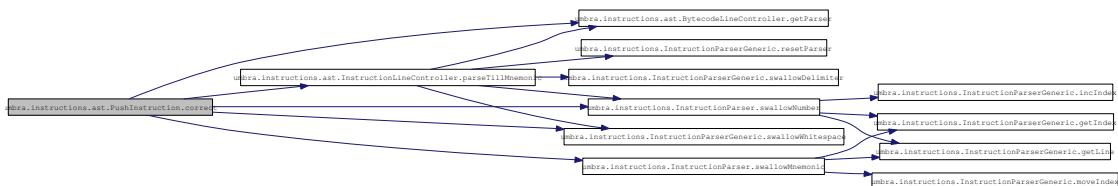
[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

References                [umbra.instructions.ast.BytocodeLineController.getParser\(\)](#),  
[umbra.instructions.ast.InstructionLineController.parseTillMnemonic\(\)](#),  
[umbra.instructions.InstructionParser.swallowMnemonic\(\)](#), [umbra.instructions.InstructionParser.swallowNumber\(\)](#),  
and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#).

Referenced by [umbra.instructions.ast.PushInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



### 6.77.3.3 int umbra.instructions.ast.PushInstruction.getInd () [protected]

This method parses the parameter of the current instruction.

This method retrieves the numerical value of the parameter of the instruction in [BytecodeLineController#getMy\\_line\\_text\(\)](#). This parameter is located after the mnemonic (with some whitespace inbetween). The method assumes [BytecodeLineController#getMy\\_line\\_text\(\)](#) is correct.

#### Returns:

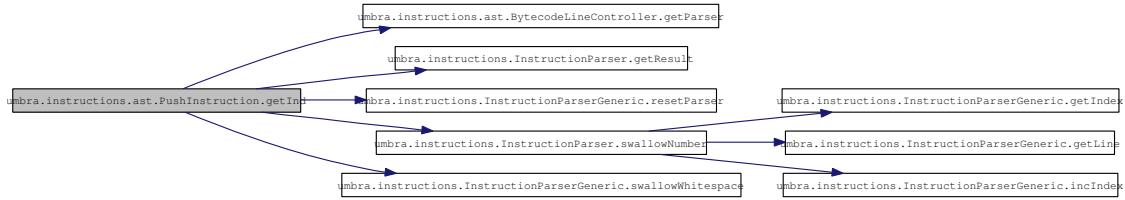
the value of the numerical parameter of the instruction

Reimplemented from [umbra.instructions.ast.MultiInstruction](#).

References                [umbra.instructions.ast.BytocodeLineController.getParser\(\)](#),  
[umbra.instructions.InstructionParser.getResult\(\)](#), [umbra.instructions.InstructionParserGeneric.resetParser\(\)](#),  
[umbra.instructions.InstructionParser.swallowNumber\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#)

Referenced by [umbra.instructions.ast.PushInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



#### 6.77.3.4 final Instruction `umbra.instructions.ast.PushInstruction.getInstruction ()`

This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for a push instruction.

It computes the parameter of the instruction before the instruction is constructed. The method can construct one of the [instructions](#):

- bipush,
- sipush.

This method also checks the syntactical correctness of the current instruction line.

#### Returns:

the freshly constructed BCEL instruction or `null` in case the instruction is not a push instruction and in case the instruction line is incorrect

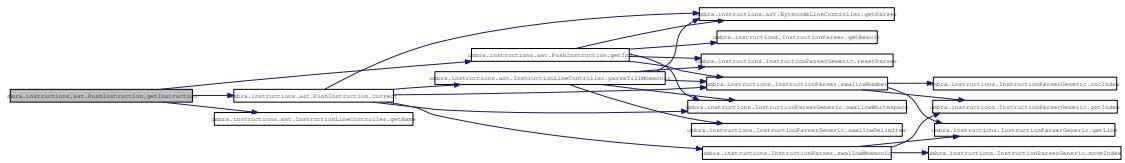
#### See also:

[BytecodeLineController.getInstruction\(\)](#)

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References [umbra.instructions.ast.PushInstruction.correct\(\)](#), [umbra.instructions.ast.PushInstruction.getInd\(\)](#), and [umbra.instructions.ast.InstructionLineController.getName\(\)](#).

Here is the call graph for this function:



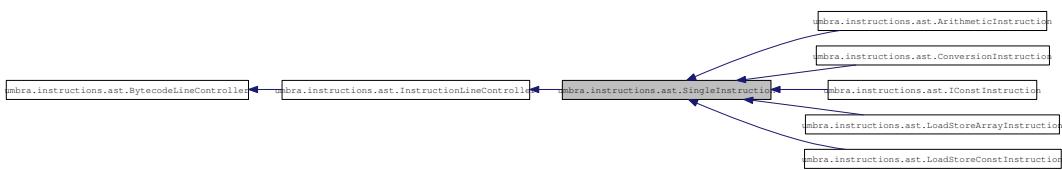
The documentation for this class was generated from the following file:

- [source/umbra/instructions/ast/PushInstruction.java](#)

## 6.78 umbra.instructions.ast.SingleInstruction Class Reference

This class handles the creation and correctness for certain [instructions](#) with no parameters.

## Inheritance diagram for `umbra.instructions.ast.SingleInstruction`:



## Collaboration diagram for `umbra.instructions.ast.SingleInstruction`:



## Public Member Functions

- `SingleInstruction` (final String a\_line\_text, final String a\_name)

This creates an instance of an instruction named as a `name` with the line text a `line`.

- Instruction `getInstruction()`

This method, based on the value of the mnemonic name, creates a new BCEL instruction object for an instruction with no parameters.

- boolean `correct()`

*Simple instruction line is correct if it has no parameter.*

## Static Public Member Functions

- static String[] getMnemonics()

This method returns the array of mnemonics for [instructions](#) with no parameters.

## Static Protected Member Functions

- static int extractConstNumber (final String a\_name, final int the\_max)

This method extracts the number from an instruction with the constant embedded in the instruction name (e.g.

- static String extractInsName (finalString a\_name)

*This method extracts the name from an instruction with the constant embedded in the instruction name (e.g.*

## Private Member Functions

- Instruction [getTopManipulationInstruction](#) (finalInstruction a\_res)  
*This method creates the objects that represent [instructions](#) that manipulate the top of the operand stack.*
- Instruction [getMonitorInstruction](#) (finalInstruction a\_res)  
*This method creates the objects that represent monitor [instructions](#).*
- Instruction [getArrayInstruction](#) (finalInstruction a\_res)  
*This method creates the objects that represent array [instructions](#).*
- Instruction [getDupInstruction](#) (finalInstruction a\_res)  
*This method creates the objects that represent dup [instructions](#).*
- Instruction [getReturnInstruction](#) (finalInstruction a\_res)  
*This method creates the objects that represent return [instructions](#).*

### 6.78.1 Detailed Description

This class handles the creation and correctness for certain [instructions](#) with no parameters.

The [instructions](#) handled here are in the following categories:

- pushing the const null value on the top of the operand stack,
- array specific [instructions](#),
- monitor [instructions](#),
- return [instructions](#),
- dup [instructions](#),
- [instructions](#) to manipulate the top of the operand stack.

#### Author:

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

### 6.78.2 Constructor & Destructor Documentation

#### 6.78.2.1 [umbra.instructions.ast.SingleInstruction.SingleInstruction](#) (final String a\_line\_text, final String a\_name)

This creates an instance of an instruction named as a\_name with the line text a\_line.

Currently it just calls the constructor of the superclass.

**Parameters:**

- a\_line\_text* the line number of the instruction
- a\_name* the mnemonic name of the instruction

**See also:**

[InstructionLineController](#).[InstructionLineController](#)(String, String)

### 6.78.3 Member Function Documentation

#### 6.78.3.1 static String [ ] [umbra.instructions.ast.SingleInstruction](#).getMnemonics () [static]

This method returns the array of mnemonics for [instructions](#) with no parameters.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController](#).[getMnemonics](#)()

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

Reimplemented in [umbra.instructions.ast.ArithmeticInstruction](#), [umbra.instructions.ast.ConversionInstruction](#), [umbra.instructions.ast.IConstInstruction](#), [umbra.instructions.ast.LoadStoreArrayInstruction](#), and [umbra.instructions.ast.LoadStoreConstInstruction](#).

#### 6.78.3.2 Instruction [umbra.instructions.ast.SingleInstruction](#).getInstruction ()

This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for an instruction with no parameters.

The method can construct an instruction from one of the following families:

- pushing the const null value on the top of the operand stack,
- array specific [instructions](#),
- monitor [instructions](#),
- return [instructions](#),
- dup [instructions](#),
- [instructions](#) to manipulate the top of the operand stack.

This method also checks the syntactical correctness of the current instruction line.

**Returns:**

the freshly constructed BCEL instruction or `null` in case the instruction is not a stack instruction and in case the instruction line is incorrect

**See also:**

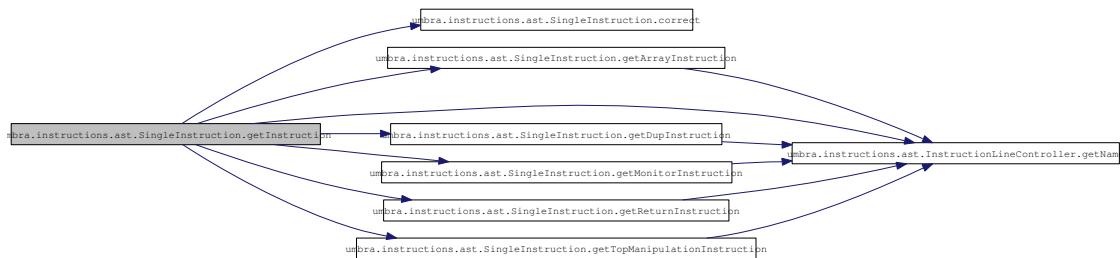
[BytecodeLineController](#).[getInstruction](#)()

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

Reimplemented in [umbra.instructions.ast.ArithmeticInstruction](#), [umbra.instructions.ast.ConversionInstruction](#), [umbra.instructions.ast.IConstInstruction](#), [umbra.instructions.ast.LoadStoreArrayInstruction](#), and [umbra.instructions.ast.LoadStoreConstInstruction](#).

References [umbra.instructions.ast.SingleInstruction.correct\(\)](#), [umbra.instructions.ast.SingleInstruction.getArrayInstruction\(\)](#), [umbra.instructions.ast.SingleInstruction.getDupInstruction\(\)](#), [umbra.instructions.ast.SingleInstruction.getMonitorInstruction\(\)](#), [umbra.instructions.ast.InstructionLineController.getName\(\)](#), [umbra.instructions.ast.SingleInstruction.getReturnInstruction\(\)](#), and [umbra.instructions.ast.SingleInstruction.getTopManipulationInstruction\(\)](#).

Here is the call graph for this function:



### 6.78.3.3 Instruction [umbra.instructions.ast.SingleInstruction.getTopManipulationInstruction](#) (final Instruction *a\_res*) [private]

This method creates the objects that represent [instructions](#) that manipulate the top of the operand stack.

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The [instructions](#) that manipulate the top of the operand stack are:

- pop,
- pop2,
- swap.

#### Parameters:

*a\_res* a helper value returned in case the current instruction is not in the current set

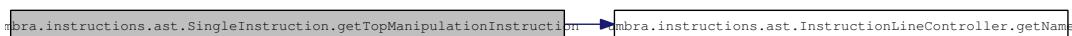
#### Returns:

the object that represents the current instruction or res in case the current instruction is not in the current set

References [umbra.instructions.ast.InstructionLineController.getName\(\)](#).

Referenced by [umbra.instructions.ast.SingleInstruction.getInstruction\(\)](#).

Here is the call graph for this function:



#### 6.78.3.4 Instruction `umbra.instructions.ast.SingleInstruction.getMonitorInstruction (final Instruction a_res) [private]`

This method creates the objects that represent monitor [instructions](#).

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The monitor [instructions](#) are:

- monitorenter,
- monitorexit.

##### Parameters:

`a_res` a helper value returned in case the current instruction is not in the current set

##### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.SingleInstruction.getInstruction()`.

Here is the call graph for this function:



#### 6.78.3.5 Instruction `umbra.instructions.ast.SingleInstruction.getArrayInstruction (final Instruction a_res) [private]`

This method creates the objects that represent array [instructions](#).

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The array [instructions](#) are:

- arraylength.

##### Parameters:

`a_res` a helper value returned in case the current instruction is not in the current set

##### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.SingleInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.78.3.6 Instruction `umbra.instructions.ast.SingleInstruction.getDupInstruction (final Instruction a_res)` [private]

This method creates the objects that represent dup [instructions](#).

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The dup [instructions](#) are:

- dup,
- dup\_x1,
- dup\_x2,
- dup2,
- dup2\_x1,
- dup2\_x2.

#### Parameters:

`a_res` a helper value returned in case the current instruction is not in the current set

#### Returns:

the object that represents the current instruction or `res` in case the current instruction is not in the current set

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.SingleInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.78.3.7 Instruction `umbra.instructions.ast.SingleInstruction.getReturnInstruction (final Instruction a_res)` [private]

This method creates the objects that represent return [instructions](#).

It checks if the name of the current instruction is one of these and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The return [instructions](#) are:

- areturn,
- dreturn,
- freturn,
- ireturn,
- lreturn,
- rreturn,
- athrow.

**Parameters:**

*a\_res* a helper value returned in case the current instruction is not in the current set

**Returns:**

the object that represents the current instruction or res in case the current instruction is not in the current set

References umbra.instructions.ast.InstructionLineController.getName().

Referenced by umbra.instructions.ast.SingleInstruction.getInstruction().

Here is the call graph for this function:



### 6.78.3.8 static int umbra.instructions.ast.SingleInstruction.extractConstNumber (final String *a\_name*, final int *the\_max*) [static, protected]

This method extracts the number from an instruction with the constant embedded in the instruction name (e.g.

iload\_0). This method additionally checks if the number does not exceed the allowed range (between 0 and max).

**Parameters:**

*a\_name* the name of the instruction

*the\_max* the maximal acceptable value of the constant

**Returns:**

the number, -1 in case the number cannot be extracted from the given name

Referenced by umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction(), and umbra.instructions.ast.IConstInstruction.getIConstInstruction().

### 6.78.3.9 static String umbra.instructions.ast.SingleInstruction.extractInsName (final String *a\_name*) [static, protected]

This method extracts the name from an instruction with the constant embedded in the instruction name (e.g.

iload\_0). This method assumes that all the sanity checks are done with the help of the method [extractConstNumber\(String, int\)](#)

#### Parameters:

*a\_name* the string with the instruction name (e.g. iload\_0)

#### Returns:

the text of the instruction name (e.g. iload for iload\_0)

Referenced by [umbra.instructions.ast.LoadStoreConstInstruction.getConstLoadStoreInstruction\(\)](#), and [umbra.instructions.ast.IConstInstruction.getIConstInstruction\(\)](#).

### 6.78.3.10 boolean umbra.instructions.ast.SingleInstruction.correct ()

Simple instruction line is correct if it has no parameter.

That means this must have the form: whitespace number : whitespace mnemonic whitespace lineend where mnemonic comes from [BytecodeStrings#SINGLE\\_INS](#).

#### Returns:

true when the instruction mnemonic is the only text in the line of the instruction text

#### See also:

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

Reimplemented in [umbra.instructions.ast.ArithmeticInstruction](#), [umbra.instructions.ast.ConversionInstruction](#), [umbra.instructions.ast.IConstInstruction](#), [umbra.instructions.ast.LoadStoreArrayInstruction](#), and [umbra.instructions.ast.LoadStoreConstInstruction](#).

Referenced by [umbra.instructions.ast.SingleInstruction.getInstruction\(\)](#).

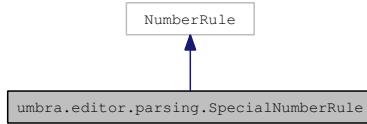
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[SingleInstruction.java](#)

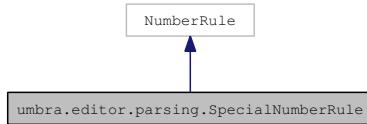
## 6.79 umbra.editor.parsing.SpecialNumberRule Class Reference

The text styling rule which extends the [NumberRule](#) so that it allows an additional mark before (and optionally after) the number to be read (used with '#' and '"').

Inheritance diagram for `umbra.editor.parsing.SpecialNumberRule`:



Collaboration diagram for `umbra.editor.parsing.SpecialNumberRule`:



### Public Member Functions

- [SpecialNumberRule](#) (final char a\_start, final char a\_end, final IToken a\_token)

*The constructor creates the rule such that the number is recognised when it is preceded with a\_start character and finished with the a\_end character.*

- [SpecialNumberRule](#) (final char a\_start, final IToken a\_token)

*The constructor creates the rule such that the number is recognised when it is preceded with a\_start character and no final character is to be checked.*

- final IToken [evaluate](#) (final ICharacterScanner a\_scanner)

*Evaluates the rule to check the number with starting and final marks.*

### Private Attributes

- char [my\\_start\\_char](#)

*The mark preceding the number.*

- char [my\\_fin](#)

*The mark after the number.*

- boolean [my\\_isfin\\_flag](#)

*The flag is true in case the final character should be checked.*

### 6.79.1 Detailed Description

The text styling rule which extends the [NumberRule](#) so that it allows an additional mark before (and optionally after) the number to be read (used with '#' and '"').

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

**Version:**

a-01

### 6.79.2 Constructor & Destructor Documentation

#### 6.79.2.1 [umbra.editor.parsing.SpecialNumberRule.SpecialNumberRule \(final char a\\_start, final char an\\_end, final IToken a\\_token\)](#)

The constructor creates the rule such that the number is recognised when it is preceded with *a\_start* character and finished with the *an\_end* character.

The token parameter is the token which is returned when the rule is successful.

**Parameters:**

*a\_start* the mark preceding the number  
*an\_end* the mark after the number  
*a\_token* the token to be returned in case the rule is successfully evaluated

**See also:**

[NumberRule.NumberRule\(IToken\)](#)

References [umbra.editor.parsing.SpecialNumberRule.my\\_isfin\\_flag](#).

#### 6.79.2.2 [umbra.editor.parsing.SpecialNumberRule.SpecialNumberRule \(final char a\\_start, final IToken a\\_token\)](#)

The constructor creates the rule such that the number is recognised when it is preceded with *a\_start* character and no final character is to be checked.

The token parameter is the token which is returned when the rule is successful.

**Parameters:**

*a\_start* the mark preceding the number  
*a\_token* the token to be returned in case the rule is successfully evaluated

**See also:**

[NumberRule.NumberRule\(IToken\)](#)

References [umbra.editor.parsing.SpecialNumberRule.my\\_isfin\\_flag](#).

### 6.79.3 Member Function Documentation

#### 6.79.3.1 final IToken umbra.editor.parsing.SpecialNumberRule.evaluate (final ICharacterScanner *a\_scanner*)

Evaluates the rule to check the number with starting and final marks.

The method scans the first character and checks if the character is the starting character of the rule. If so, it swallows a number. If the scanning of the number is successful the method checks if it must check the final character. If not, it returns successfully. If so it checks the final character. In case it matches the proper final character the method returns successfully. Otherwise, it puts back the final character, the characters of the number and the starting character to the scanner.

The token returned by this rule returns `true` when calling `isUndefined`, if the text that the rule investigated does not match the rule's requirements.

**Parameters:**

*a\_scanner* the character scanner to be used to obtain the token

**Returns:**

the recognised token (supplied in the constructor) or `Token#UNDEFINED` in case the rule does not apply

**See also:**

`NumberRule.evaluate(ICharacterScanner)`

References `umbra.editor.parsing.SpecialNumberRule.my_fin`, `umbra.editor.parsing.SpecialNumberRule.my_isfin_flag`, and `umbra.editor.parsing.SpecialNumberRule.my_start_char`.

### 6.79.4 Member Data Documentation

#### 6.79.4.1 char umbra.editor.parsing.SpecialNumberRule.my\_start\_char [private]

The mark preceding the number.

Referenced by `umbra.editor.parsing.SpecialNumberRule.evaluate()`.

#### 6.79.4.2 char umbra.editor.parsing.SpecialNumberRule.my\_fin [private]

The mark after the number.

Referenced by `umbra.editor.parsing.SpecialNumberRule.evaluate()`.

#### 6.79.4.3 boolean umbra.editor.parsing.SpecialNumberRule.my\_isfin\_flag [private]

The flag is `true` in case the final character should be checked.

Referenced by `umbra.editor.parsing.SpecialNumberRule.evaluate()`, `umbra.editor.parsing.SpecialNumberRule.SpecialNumberRule()`, and `umbra.editor.parsing.SpecialNumberRule()`.

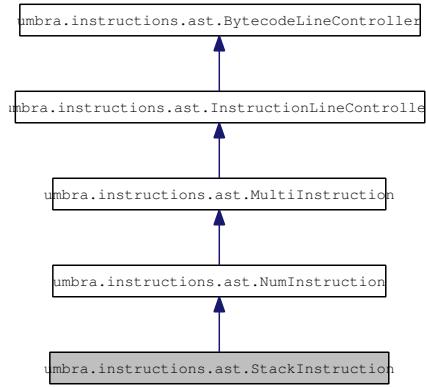
The documentation for this class was generated from the following file:

- `source/umbra/editor/parsing/SpecialNumberRule.java`

## 6.80 umbra.instructions.ast.StackInstruction Class Reference

This class handles the creation and correctness for load and store [instructions](#) with parameters i.e.

Inheritance diagram for `umbra.instructions.ast.StackInstruction`:



Collaboration diagram for `umbra.instructions.ast.StackInstruction`:



### Public Member Functions

- `StackInstruction (final String a_line_text, final String a_name)`

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line.*

- `final boolean correct ()`

*Check the correctness of a stack instruction line.*

- `final Instruction getInstruction ()`

*This method, based on the value of the the mnemonic name, creates a new BCEL instruction object for a stack instruction.*

### Static Public Member Functions

- `static String[] getMnemonics ()`

*This method returns the array of stack [instructions](#) mnemonics.*

### Protected Member Functions

- `int getInd ()`

*This method parses the parameter of the current instruction.*

## Private Member Functions

- Instruction [getLInstruction](#) (final int an\_index, finalInstruction a\_res)  
*This method creates the objects that represents l-instructions.*
- Instruction [getIInstruction](#) (final int an\_index, finalInstruction a\_res)  
*This method creates the objects that represents i-instructions.*
- Instruction [getFInstruction](#) (final int an\_index, finalInstruction a\_res)  
*This method creates the objects that represents f-instructions.*
- Instruction [getDInstruction](#) (final int an\_index, finalInstruction a\_res)  
*This method creates the objects that represents d-instructions.*
- Instruction [getAInstruction](#) (final int an\_index, finalInstruction a\_res)  
*This method creates the objects that represents a-instructions.*

### 6.80.1 Detailed Description

This class handles the creation and correctness for load and store [instructions](#) with parameters i.e.

:

- aload,
- astore,
- dload,
- dstore,
- fload,
- fstore,
- iload,
- istore,
- lload,
- lstore.

#### Author:

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Aleksy Schubert ([alx@mimuw.edu.pl](mailto:alx@mimuw.edu.pl))

#### Version:

a-01

## 6.80.2 Constructor & Destructor Documentation

### 6.80.2.1 **umbra.instructions.ast.StackInstruction.StackInstruction (final String *a\_line\_text*, final String *a\_name*)**

This creates an instance of an instruction named as *a\_name* in a line the text of which is *a\_line*.

Currently it just calls the constructor of the superclass.

#### Parameters:

*a\_line\_text* the line number of the instruction

*a\_name* the mnemonic name of the instruction

#### See also:

[InstructionLineControllerInstructionLineController\(String, String\)](#)

## 6.80.3 Member Function Documentation

### 6.80.3.1 **static String [ ] umbra.instructions.ast.StackInstruction.getMnemonics () [static]**

This method returns the array of stack [instructions](#) mnemonics.

#### Returns:

the array of the handled mnemonics

#### See also:

[InstructionLineController.getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

### 6.80.3.2 **final boolean umbra.instructions.ast.StackInstruction.correct ()**

Check the correctness of a stack instruction line.

The line is correct when it has the form: whitespace number : whitespace mnemonic whitespace % whitespace number whitespace lineend

#### Returns:

true when the syntax of the instruction line is correct

#### See also:

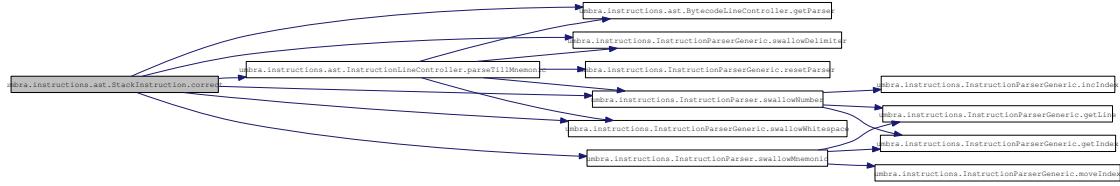
[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

References                  [umbra.instructions.ast.BytocodeLineController.getParser\(\)](#),  
[umbra.instructions.ast.InstructionLineController.parseTillMnemonic\(\)](#),  
[umbra.instructions.InstructionParserGeneric.swallowDelimiter\(\)](#), [umbra.instructions.InstructionParser.swallowMnemonic\(\)](#),  
[umbra.instructions.InstructionParser.swallowNumber\(\)](#), and [umbra.instructions.InstructionParserGeneric.swallowWhitespace\(\)](#)

Referenced by `umbra.instructions.ast.StackInstruction.getInstruction()`.

Here is the call graph for this function:



#### **6.80.3.3 int umbra.instructions.ast.StackInstruction.getInd () [protected]**

This method parses the parameter of the current instruction.

This method retrieves the numerical value of the index parameter of the instruction in [BytecodeLineController#getMy\\_line\\_text\(\)](#). This parameter is located after the first " character in the line. The method assumes [BytecodeLineController#getMy\\_line\\_text\(\)](#) is correct.

## Returns:

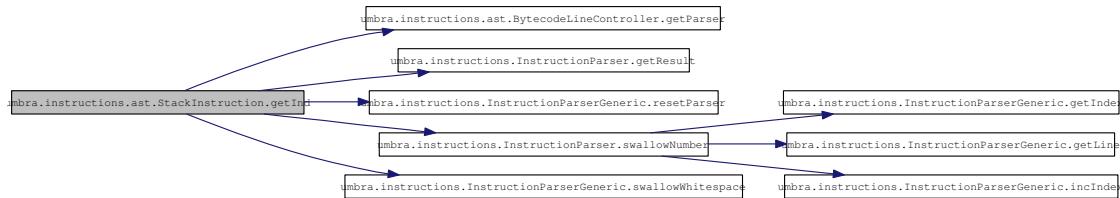
the value of the numerical parameter of the instruction

Reimplemented from [umbra.instructions.ast.MultiInstruction](#).

References `umbra.instructions.ast.BytecodeLineController.getParser()`, `umbra.instructions.InstructionParser.getResult()`, `umbra.instructions.InstructionParserGeneric.resetParser()`, `umbra.instructions.InstructionParser.swallowNumber()`, and `umbra.instructions.InstructionParserGeneric.swallowNumber()`.

Referenced by `umbra.instructions.ast.StackInstruction.getInstruction()`.

Here is the call graph for this function:



#### 6.80.3.4 final Instruction umbra.instructions.ast.StackInstruction.getInstruction()

This method, based on the value of the mnemonic name, creates a new BCEL instruction object for a stack instruction.

It computes the index parameter of the instruction before the instruction is constructed. The method can construct one of the [instructions](#):

- `aload`,
  - `astore`.

- dload,
  - dstore,
  - fload,
  - fstore,
  - iload,
  - istore,
  - lload,
  - lstore.

This method also checks the syntactical correctness of the current instruction line.

## Returns:

the freshly constructed BCEL instruction or null in case the instruction is not a stack instruction and in case the instruction line is incorrect

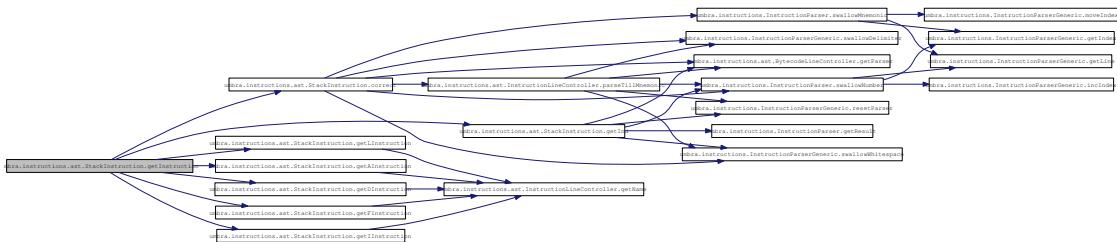
#### **See also:**

## BytecodeLineController.getInstruction()

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

References `umbra.instructions.ast.StackInstruction.correct()`, `umbra.instructions.ast.StackInstruction.getAInstruction()`, `umbra.instructions.ast.StackInstruction.getDInstruction()`, `umbra.instructions.ast.StackInstruction.getFInstruction()`, `umbra.instructions.ast.StackInstruction.getIIInstruction()`, `umbra.instructions.ast.StackInstruction.getInd()`, and `umbra.instructions.ast.StackInstruction.getLInstruction()`.

Here is the call graph for this function:



#### **6.80.3.5 Instruction umbra.instructions.ast.StackInstruction.getLInstruction (final int *an\_index*, final Instruction *a\_res*) [private]**

This method creates the objects that represents l-instructions.

It checks if the name of the current instruction is one of the l-instructions and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The l-instructions are:

- lload,

- istore.

**Parameters:**

*an\_index* the parameter of the instruction to be created

*a\_res* a helper value returned in case the current instruction is not an l-instruction

**Returns:**

the object that represents the current l-instruction or res in case the current instruction is not an l-instruction

References umbra.instructions.ast.InstructionLineController.getName().

Referenced by umbra.instructions.ast.StackInstruction.getInstruction().

Here is the call graph for this function:



### 6.80.3.6 Instruction umbra.instructions.ast.StackInstruction.getIInstruction (final int *an\_index*, final Instruction *a\_res*) [private]

This method creates the objects that represents i-instructions.

It checks if the name of the current instruction is one of the i-instructions and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter *a\_res*.

The i-instructions are:

- iload,
- istore.

**Parameters:**

*an\_index* the parameter of the instruction to be created

*a\_res* a helper value returned in case the current instruction is not an i-instruction

**Returns:**

the object that represents the current i-instruction or res in case the current instruction is not an i-instruction

**See also:**

[BytecodeLineController.getInstruction\(\)](#)

References umbra.instructions.ast.InstructionLineController.getName().

Referenced by umbra.instructions.ast.StackInstruction.getInstruction().

Here is the call graph for this function:



### 6.80.3.7 Instruction `umbra.instructions.ast.StackInstruction.getFInstruction (final int an_index, final Instruction a_res)` [private]

This method creates the objects that represents f-instructions.

It checks if the name of the current instruction is one of the f-instructions and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The f-instructions are:

- fload,
- fstore.

#### Parameters:

`an_index` the parameter of the instruction to be created

`a_res` a helper value returned in case the current instruction is not an f-instruction

#### Returns:

the object that represents the current f-instruction or `res` in case the current instruction is not an f-instruction

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.StackInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.80.3.8 Instruction `umbra.instructions.ast.StackInstruction.getDInstruction (final int an_index, final Instruction a_res)` [private]

This method creates the objects that represents d-instructions.

It checks if the name of the current instruction is one of the d-instructions and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The d-instructions are:

- dload,
- dstore.

#### Parameters:

`an_index` the parameter of the instruction to be created

`a_res` a helper value returned in case the current instruction is not an d-instruction

#### Returns:

the object that represents the current d-instruction or `res` in case the current instruction is not a d-instruction

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.StackInstruction.getInstruction()`.

Here is the call graph for this function:



### 6.80.3.9 Instruction `umbra.instructions.ast.StackInstruction.getAInstruction (final int an_index, final Instruction a_res)` [private]

This method creates the objects that represents a-instructions.

It checks if the name of the current instruction is one of the a-instructions and in that case creates an appropriate object. In case the name is of a different kind it returns the parameter `a_res`.

The a-instructions are:

- `aload`,
- `astore`.

#### Parameters:

`an_index` the parameter of the instruction to be created

`a_res` a helper value returned in case the current instruction is not an a-instruction

#### Returns:

the object that represents the current a-instruction or `res` in case the current instruction is not an a-instruction

References `umbra.instructions.ast.InstructionLineController.getName()`.

Referenced by `umbra.instructions.ast.StackInstruction.getInstruction()`.

Here is the call graph for this function:



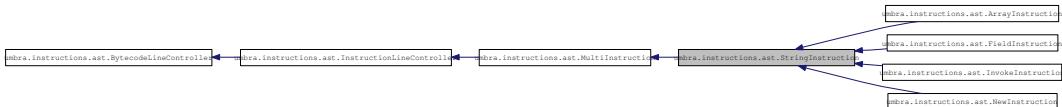
The documentation for this class was generated from the following file:

- `source/umbra/instructions/ast/StackInstruction.java`

## 6.81 umbra.instructions.ast.StringInstruction Class Reference

This is abstract class for all [instructions](#) with a string as a parameter.

Inheritance diagram for `umbra.instructions.ast.StringInstruction`:



Collaboration diagram for `umbra.instructions.ast.StringInstruction`:



### Public Member Functions

- `StringInstruction (final String a_line_text, final String a_name)`

*This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.*

#### 6.81.1 Detailed Description

This is abstract class for all [instructions](#) with a string as a parameter.

##### Author:

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

##### Version:

a-01

#### 6.81.2 Constructor & Destructor Documentation

##### 6.81.2.1 `umbra.instructions.ast.StringInstruction (final String a_line_text, final String a_name)`

This creates an instance of an instruction named as a\_name in a line the text of which is a\_line\_text.

Currently it just calls the constructor of the superclass.

##### Parameters:

- `a_line_text` the line number of the instruction
- `a_name` the mnemonic name of the instruction

##### See also:

`InstructionLineControllerInstructionLineController(String, String)`

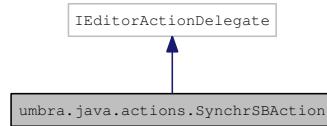
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[StringInstruction.java](#)

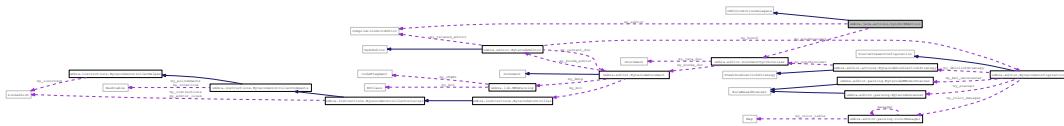
## 6.82 umbra.java.actions.SynchrSBAction Class Reference

This class defines an action of synchronization positions from source code to bytecode.

Inheritance diagram for umbra.java.actions.SynchrSBAction:



Collaboration diagram for umbra.java.actions.SynchrSBAction:



### Public Member Functions

- final void `setActiveEditor` (final IAction an\_action, final IEditorPart a\_java\_editor)  
*The method sets the internal Java source code `editor` attribute.*
- final void `run` (final IAction an\_action)  
*This method handles the action of the synchronisation between the source code and the byte code i.e.*
- void `selectionChanged` (final IAction an\_action, final ISelection a\_selection)  
*Currently, does nothing.*

### Private Member Functions

- void `synchronizeWithMessages` (final int an\_offset, final BytecodeDocument a\_bcode\_doc)  
*This method performs the synchronisation of the byte code document for the given position in the source code document.*
- `DocumentSynchroniser getDocSynch` (final BytecodeDocument a\_doc)  
*This method lazily provides the object which performs the synchronisation operations.*

### Private Attributes

- `CompilationUnitEditor my_editor`  
*The `editor` of the Java source code.*
- `DocumentSynchroniser my_synchroniser`  
*This is an object which handles the calculations of the synchronisation positions.*

### 6.82.1 Detailed Description

This class defines an action of synchronization positions form source code to bytecode.

It is available with the standard Java [editor](#).

**Author:**

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

**Version:**

a-01

### 6.82.2 Member Function Documentation

#### 6.82.2.1 final void umbra.java.actions.SynchrSBAction.setActiveEditor (final IAction *an\_action*, final IEeditorPart *a\_java\_editor*)

The method sets the internal Java source code [editor](#) attribute.

**Parameters:**

*an\_action* the action which triggered the change of the [editor](#)

*a\_java\_editor* the new Java source code [editor](#) to be associated with the action

References [umbra.java.actions.SynchrSBAction.my\\_editor](#).

#### 6.82.2.2 final void umbra.java.actions.SynchrSBAction.run (final IAction *an\_action*)

This method handles the action of the synchronisation between the source code and the byte code i.e.

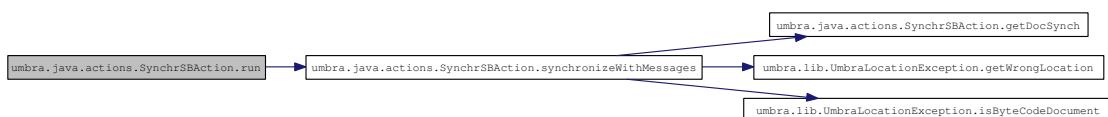
it takes the selection in the source code and shows the corresponding selection in the byte code.

**Parameters:**

*an\_action* the action that triggered the operation

References [umbra.java.actions.SynchrSBAction.my\\_editor](#), [umbra.java.actions.SynchrSBAction.synchronizeWithMessages\(\)](#). and [umbra.lib.UmbralocationException.getWrongLocation](#)

Here is the call graph for this function:



#### 6.82.2.3 void umbra.java.actions.SynchrSBAction.synchronizeWithMessages (final int *an\_offset*, final BytecodeDocument *a\_bcode\_doc*) [private]

This method performs the synchronisation of the byte code document for the given position in the source code document.

This method additionally pops up all the necessary messages in case exceptions are raised.

**Parameters:**

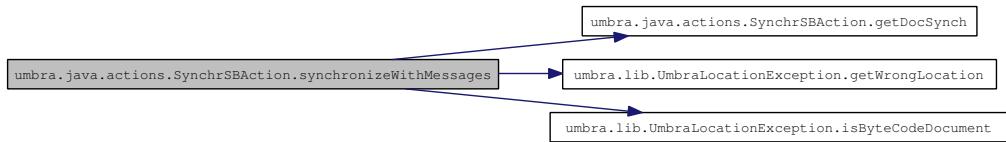
*an\_offset* a position in the source code [editor](#). Lines in related byte code [editor](#) containing the line with this position will be highlighted

*a\_bcode\_doc* the byte code document to synchronise

References `umbra.java.actions.SynchrSBAction.getDocSynch()`, `umbra.lib.UmbralocationException.getWrongLocation()`, `umbra.lib.UmbralocationException.isByteCodeDocument()`, and `umbra.java.actions.SynchrSBAction.my_editor`.

Referenced by `umbra.java.actions.SynchrSBAction.run()`.

Here is the call graph for this function:



#### 6.82.2.4 void `umbra.java.actions.SynchrSBAction.selectionChanged (final IAction an_action, final ISelection a_selection)`

Currently, does nothing.

**Parameters:**

*an\_action* see [ISelection](#))

*a\_selection* see [ISelection](#))

#### 6.82.2.5 DocumentSynchroniser `umbra.java.actions.SynchrSBAction.getDocSynch (final BytecodeDocument a_doc) [private]`

This method lazily provides the object which performs the synchronisation operations.

**Parameters:**

*a\_doc* a byte code document for which the synchronisation is performed

**Returns:**

a [DocumentSynchroniser](#) which performs the synchronisation operations

References `umbra.java.actions.SynchrSBAction.my_editor`, and `umbra.java.actions.SynchrSBAction.my_synchroniser`.

Referenced by `umbra.java.actions.SynchrSBAction.synchronizeWithMessages()`.

### 6.82.3 Member Data Documentation

#### 6.82.3.1 CompilationUnitEditor `umbra.java.actions.SynchrSBAction.my_editor [private]`

The [editor](#) of the Java source code.

Referenced by umbra.java.actions.SynchrSBAction.getDocSynch(),  
umbra.java.actions.SynchrSBAction.run(), umbra.java.actions.SynchrSBAction.setActiveEditor(),  
umbra.java.actions.SynchrSBAction.synchronizeWithMessages(). um-  
and

### 6.82.3.2 DocumentSynchroniser umbra.java.actions.SynchrSBAction.my\_synchroniser [private]

This is an object which handles the calculations of the synchronisation positions.

Referenced by umbra.java.actions.SynchrSBAction.getDocSynch().

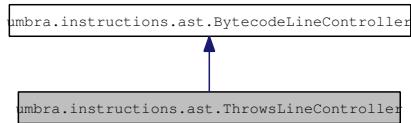
The documentation for this class was generated from the following file:

- source/umbra/java/actions/[SynchrSBAction.java](#)

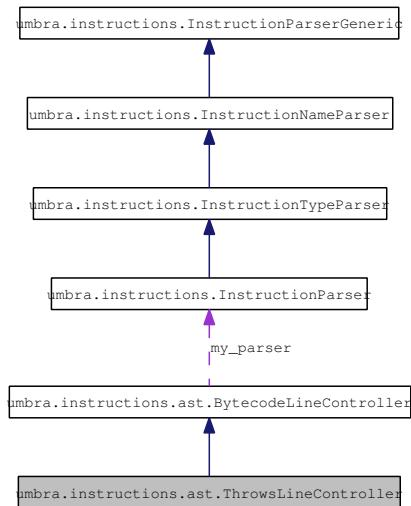
## 6.83 umbra.instructions.ast.ThrowsLineController Class Reference

This is a class for a special bytecode lines related to thrown exceptions, not to be edited by a user.

Inheritance diagram for umbra.instructions.ast.ThrowsLineController:



Collaboration diagram for umbra.instructions.ast.ThrowsLineController:



### Public Member Functions

- [ThrowsLineController \(final String a\\_line\\_text\)](#)  
*This constructor remembers only the line text of a line with the throws instruction.*
- final boolean [correct \(\)](#)  
*Checks the correctness of throws lines.*

#### 6.83.1 Detailed Description

This is a class for a special bytecode lines related to thrown exceptions, not to be edited by a user.

##### Author:

Tomek Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

##### Version:

a-01

### 6.83.2 Constructor & Destructor Documentation

#### 6.83.2.1 umbra.instructions.ast.ThrowsLineController.ThrowsLineController (final String *a\_line\_text*)

This constructor remembers only the line text of a line with the throws instruction.

**Parameters:**

*a\_line\_text* the string representation of the line

**See also:**

BytecodeLineController.BytecodeLineController(String)

### 6.83.3 Member Function Documentation

#### 6.83.3.1 final boolean umbra.instructions.ast.ThrowsLineController.correct ()

Checks the correctness of throws lines.

Currently, the correctness of this kind of line is handled in a very crude way. This is due to the fact that the bytecode textual representation has no throws lines for the time being.

**Returns:**

currently, it returns always true

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[ThrowsLineController.java](#)

## 6.84 umbra.editor.parsing.TokenGetter Class Reference

This is an intermediary class which creates the Eclipse [parsing](#) and text partitioning classes with the properties established using the Umbra colouring modes.

### Static Public Member Functions

- static IToken [getToken](#) (final ColorManager the\_colour\_manager, final int a\_mode, final int a\_col)  
*Returns a fresh token with associated colour.*
- static IToken[ ] [getTokenTypeTab](#) (final ColorManager the\_manager, final int a\_mode)  
*Returns the array with tokens for all the possible areas in the BML documents.*
- static NonRuleBasedDamagerRepairer [getRepairer](#) (final ColorManager a\_manager, final int a\_mode, final int a\_col)  
*Returns a fresh damager-repairer that determines the damaged region and creates the presentation using the given colour in the given colouring mode with the given colour manager.*

### Private Member Functions

- [TokenGetter \(\)](#)  
*This is a utility class so we declare a private constructor to prevent accidental creation of the instances.*

### Static Private Member Functions

- static TextAttribute [getTextAttribute](#) (final ColorManager the\_manager, final int a\_mode, final int a\_col)  
*Creates a text attribute for the given colour manager, colouring mode and the colour number.*

### 6.84.1 Detailed Description

This is an intermediary class which creates the Eclipse [parsing](#) and text partitioning classes with the properties established using the Umbra colouring modes.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

#### Version:

a-01

### 6.84.2 Constructor & Destructor Documentation

#### 6.84.2.1 umbra.editor.parsing.TokenGetter () [private]

This is a utility class so we declare a private constructor to prevent accidental creation of the instances.

### 6.84.3 Member Function Documentation

#### 6.84.3.1 static ITok en umbra.editor.parsing.TokenGetter.getToken (final ColorManager the\_colour\_manager, final int a\_mode, final int a\_col) [static]

Returns a fresh token with associated colour.

The colour is retrieved from the given colour manager and is computed based on the given colouring mode and the colour number within the mode.

##### Parameters:

*the\_colour\_manager* the colour manager related to the current byte code [editor](#), it must be the same as in the current [umbra.editor.BytecodeConfiguration](#) object

*a\_mode* the number of the current colouring style, it must be the same as in the current [umbra.editor.BytecodeConfiguration](#) object

*a\_col* a colour value with fixed meaning across the colouring styles

##### Returns:

the colour value as a token

References [umbra.editor.parsing.TokenGetter.getTextAttribute\(\)](#).

Here is the call graph for this function:



#### 6.84.3.2 static ITok en [] umbra.editor.parsing.TokenGetter.getTokenTab (final ColorManager the\_manager, final int a\_mode) [static]

Returns the array with tokens for all the possible areas in the BML documents.

##### Parameters:

*the\_manager* the colour manager related to the current byte code [editor](#), it must be the same as the one in the current [umbra.editor.BytecodeConfiguration](#) object

*a\_mode* the number of the current colouring style, it must be the same as in the current [umbra.editor.BytecodeConfiguration](#) object

##### Returns:

array of tokens - one for each area

#### 6.84.3.3 static NonRuleBasedDamagerRepairer umbra.editor.parsing.TokenGetter.getRepairer (final ColorManager a\_manager, final int a\_mode, final int a\_col) [static]

Returns a fresh damager-repairer that determines the damaged region and creates the presentation using the given colour in the given colouring mode with the given colour manager.

**Parameters:**

*a\_manager* manager the colour manager related to the current byte code *editor*, it must be the same as in the current `umbra.editor.BytecodeConfiguration` object

*a\_mode* the number of the current colouring style, it must be the same as in the current `umbra.editor.BytecodeConfiguration` object

*a\_col* particular abstract colour as an attribute

**Returns:**

each time a new damage repairer with the given colour parameters

References `umbra.editor.parsing.TokenGetter.getTextAttribute()`.

Here is the call graph for this function:



#### 6.84.3.4 static TextAttribute `umbra.editor.parsing.TokenGetter.getTextAttribute (final ColorManager the_manager, final int a_mode, final int a_col)` [static, private]

Creates a text attribute for the given colour manager, colouring mode and the colour number.

The returned `TextAttribute` has the foreground colour set according to the `ColorValues#MODES_DESC` array, the background colour set to be the default and the style again set according to the `ColorValues#MODES_DESC`.

**Parameters:**

*the\_manager* the colour manager related to the current byte code *editor*, it must be the same as in the current `umbra.editor.BytecodeConfiguration` object

*a\_mode* the number of the current colouring style, it must be the same as in the current `umbra.editor.BytecodeConfiguration` object

*a\_col* a colour value with fixed meaning across all the colouring styles

**Returns:**

the given colour as an attribute

References `umbra.editor.parsing.ColorManager.getColor()`.

Referenced by `umbra.editor.parsing.TokenGetter.getRepairer()`, and `umbra.editor.parsing.TokenGetter.getToken()`.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- source/umbra/editor/parsing/`TokenGetter.java`

## 6.85 umbra.lib.UmbraClassException Class Reference

This is an exception used to trace situations when a problem with class file is encountered.

### Public Member Functions

- [UmbraClassException](#) (final Exception *an\_exception*)  
*Creates an exception with the exception that caused the current one.*
- Exception [getCause](#) ()  
*Returns the exception which caused the current one.*

### Private Attributes

- final Exception [my\\_exception](#)  
*This field contains the exception which triggered the current one.*

### Static Private Attributes

- static final long [serialVersionUID](#) = 1344810623005640402L  
*The serial number of the class.*

### 6.85.1 Detailed Description

This is an exception used to trace situations when a problem with class file is encountered.

It is used to encapsulate [ClassNotFoundException](#) or [annot.io.ReadAttributeException](#)

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.85.2 Constructor & Destructor Documentation

#### 6.85.2.1 umbra.lib.UmbraClassException.UmbraClassException (final Exception *an\_exception*)

Creates an exception with the exception that caused the current one.

#### Parameters:

*an\_exception* the exception which caused the current one

References [umbra.lib.UmbraClassException.my\\_exception](#).

### 6.85.3 Member Function Documentation

#### 6.85.3.1 Exception umbra.lib.UmbraClassException.getCause ()

Returns the exception which caused the current one.

**Returns:**

the exception which caused the current one

References umbra.lib.UmbraClassException.my\_exception.

Referenced by umbra.editor.actions.BytecodeCombineAction.updateMethods().

### 6.85.4 Member Data Documentation

#### 6.85.4.1 final long umbra.lib.UmbraClassException serialVersionUID = 1344810623005640402L [static, private]

The serial number of the class.

#### 6.85.4.2 final Exception umbra.lib.UmbraClassException.my\_exception [private]

This field contains the exception which triggered the current one.

Referenced by umbra.lib.UmbraClassException.getCause(), and umbra.lib.UmbraClassException.UmbraClassException().

The documentation for this class was generated from the following file:

- source/umbra/lib/[UmbraClassException.java](#)

## 6.86 umbra.lib.UmbraException Class Reference

This is an exception used in tracing internal exceptional flows within Umbra.

### Public Member Functions

- [UmbraException \(\)](#)  
*The standard way to create the exception.*

### Static Private Attributes

- static final long [serialVersionUID = -8982650711998004110L](#)  
*The serial ID for the exception.*

### 6.86.1 Detailed Description

This is an exception used in tracing internal exceptional flows within Umbra.

This exception should not be handled outside Umbra.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.86.2 Constructor & Destructor Documentation

#### 6.86.2.1 umbra.lib.UmbraException.UmbraException ()

The standard way to create the exception.

### 6.86.3 Member Data Documentation

#### 6.86.3.1 final long umbra.lib.UmbraException.serialVersionUID = -8982650711998004110L [static, private]

The serial ID for the exception.

The documentation for this class was generated from the following file:

- source/umbra/lib/[UmbraException.java](#)

## 6.87 umbra.lib.UmbraLocationException Class Reference

This is an exception used to trace situations when the parsing exceeded the content of a document.

### Public Member Functions

- [UmbraLocationException](#) (final int a\_loc, final boolean a\_line)  
*Creates an exception with the information that the number of the line outside the document.*
- [UmbraLocationException](#) (final boolean a\_doc\_type, final int a\_loc, final boolean a\_line)  
*Creates exception with the information that the number of the line outside the document and a document type.*
- int [getWrongLocation](#) ()  
*Returns the number of the wrong line.*
- boolean [isByteCodeDocument](#) ()  
*Retruns true in case the editor is a byte code editor.*
- boolean [isLineWrong](#) ()  
*Returns information on how to interpret the wrong location number.*

### Private Attributes

- final int [my\\_wrong\\_location](#)  
*This field contains the location which is considered to be wrong.*
- final boolean [my\\_islinewrong](#)  
*This field is true in case the wrong location is to be interpreted as a line.*
- final boolean [my\\_doc\\_type](#)  
*This field is true in case the document is a byte code document.*

### Static Private Attributes

- static final long [serialVersionUID](#) = 1368987676616348613L  
*The serial number of the class.*

#### 6.87.1 Detailed Description

This is an exception used to trace situations when the parsing exceeded the content of a document.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

**Version:**

a-01

## 6.87.2 Constructor & Destructor Documentation

### 6.87.2.1 umbra.lib.UmbraLocationException.UmbraLocationException (final int *a\_loc*, final boolean *a\_line*)

Creates an exception with the information that the number of the line outside the document.

We assume that the document type here is byte code document.

**Parameters:**

*a\_loc* the location outside the document

*a\_line* true when the location is a line number, false when the location is a position number

References umbra.lib.UmbraLocationException.my\_doc\_type, umbra.lib.UmbraLocationException.my\_islinewrong, and umbra.lib.UmbraLocationException.my\_wrong\_location.

### 6.87.2.2 umbra.lib.UmbraLocationException.UmbraLocationException (final boolean *a\_doc\_type*, final int *a\_loc*, final boolean *a\_line*)

Creates exception with the information that the number of the line outside the document and a document type.

We check two kinds of documents: byte code documents and Java source code documents.

**Parameters:**

*a\_doc\_type* the type of document for which the location exception is thrown: true for [umbra.editor.BytecodeDocument](#) and false for [org.eclipse.jdt.internal.ui.javaeditor.CompilationUnitEditor](#)

*a\_loc* the location outside the document

*a\_line* true when the location is a line number, false when the location is a position number

References umbra.lib.UmbraLocationException.my\_doc\_type, umbra.lib.UmbraLocationException.my\_islinewrong, and umbra.lib.UmbraLocationException.my\_wrong\_location.

## 6.87.3 Member Function Documentation

### 6.87.3.1 int umbra.lib.UmbraLocationException.getWrongLocation ()

Returns the number of the wrong line.

**Returns:**

the number of the wrong line

References umbra.lib.UmbraLocationException.my\_wrong\_location.

Referenced by [umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged\(\)](#), [umbra.lib.GUIMessages.exceededRangeInfo\(\)](#), [umbra.editor.parsing.NonRuleBasedDamagerRepairer.getDamageRegion\(\)](#),

umbra.editor.actions.history.BytecodeRestoreAction.refreshContent(), umbra.editor.actions.BytecodeColorAction.run(), umbra.java.actions.SynchrSBAction.synchronizeWithMessages(), umbra.editor.BytecodeContribution.BytecodeListener.updateFragment(), and umbra.editor.actions.BytecodeEditorAction.wrongLocationMessage().

#### 6.87.3.2 boolean umbra.lib.UmbraLocationException.isByteCodeDocument ()

Retruns true in case the editor is a byte code editor.

**Returns:**

true in case the editor is a byte code editor, false otherwise

References umbra.lib.UmbraLocationException.my\_doc\_type.

Referenced by umbra.java.actions.SynchrSBAction.synchronizeWithMessages(), and umbra.editor.actions.BytecodeEditorAction.wrongLocationMessage().

#### 6.87.3.3 boolean umbra.lib.UmbraLocationException.isLineWrong ()

Returns information on how to interpret the wrong location number.

**Returns:**

true when the wrong location number is to be interpreted as a line number, otherwise the location number is to be interpreted as a position in a document

References umbra.lib.UmbraLocationException.my\_islinewrong.

Referenced by umbra.editor.parsing.NonRuleBasedDamagerRepairer.getDamageRegion().

### 6.87.4 Member Data Documentation

#### 6.87.4.1 final long umbra.lib.UmbraLocationException.serialVersionUID = 1368987676616348613L [static, private]

The serial number of the class.

#### 6.87.4.2 final int umbra.lib.UmbraLocationException.my\_wrong\_location [private]

This field contains the location which is considered to be wrong.

Referenced by umbra.lib.UmbraLocationException.getWrongLocation(), and umbra.lib.UmbraLocationException.UmbraLocationException().

#### 6.87.4.3 final boolean umbra.lib.UmbraLocationException.my\_islinewrong [private]

This field is true in case the wrong location is to be interpreted as a line.

In case it is false, the wrong location is a position in the document.

Referenced by umbra.lib.UmbraLocationException.isLineWrong(), and umbra.lib.UmbraLocationException.UmbraLocationException().

**6.87.4.4 final boolean umbra.lib.UmbraLocationException.my\_doc\_type [private]**

This field is `true` in case the document is a byte code document.

Otherwise, the document is a Java source code document.

Referenced by `umbra.lib.UmbraLocationException.isByteCodeDocument()`, and `umbra.lib.UmbraLocationException.UmbraLocationException()`.

The documentation for this class was generated from the following file:

- `source/umbra/lib/UmbraLocationException.java`

## 6.88 umbra.lib.UmbraMethodException Class Reference

This is an exception used to trace situations when the processing reached a method which does not exist in the document being processed.

### Public Member Functions

- [UmbraMethodException \(final int a\\_mno\)](#)  
*Creates an exception with the information on the number of the method outside the document.*
- [int getWrongMethodNumber \(\)](#)  
*Returns the number of the wrong method.*

### Private Attributes

- [final int my\\_wrong\\_methodno](#)  
*This field contains the method number which is considered to be wrong.*

### Static Private Attributes

- [static final long serialVersionUID = 5973766671008411853L](#)  
*The serial number of the class.*

#### 6.88.1 Detailed Description

This is an exception used to trace situations when the processing reached a method which does not exist in the document being processed.

##### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

##### Version:

a-01

#### 6.88.2 Constructor & Destructor Documentation

##### 6.88.2.1 umbra.lib.UmbraMethodException.UmbraMethodException (final int a\_mno)

Creates an exception with the information on the number of the method outside the document.

##### Parameters:

*a\_mno* the method number outside the document

References [umbra.lib.UmbraMethodException.my\\_wrong\\_methodno](#).

### 6.88.3 Member Function Documentation

#### 6.88.3.1 int umbra.lib.UmbraMethodException.getWrongMethodNumber ()

Returns the number of the wrong method.

##### Returns:

the number of the wrong method

References umbra.lib.UmbraMethodException.my\_wrong\_methodno.

Referenced by umbra.editor.BytecodeContribution.BytecodeListener.documentAboutToBeChanged(), umbra.lib.GUIMessages.exceededRangeInfo(), umbra.editor.actions.history.BytecodeRestoreAction.refreshContent(), and umbra.editor.actions.BytecodeColorAction.run().

### 6.88.4 Member Data Documentation

#### 6.88.4.1 final long umbra.lib.UmbraMethodException serialVersionUID = 5973766671008411853L [static, private]

The serial number of the class.

#### 6.88.4.2 final int umbra.lib.UmbraMethodException.my\_wrong\_methodno [private]

This field contains the method number which is considered to be wrong.

Referenced by umbra.lib.UmbraMethodException.getWrongMethodNumber(), and umbra.lib.UmbraMethodException.UmbraMethodException().

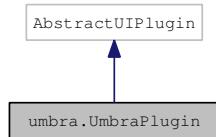
The documentation for this class was generated from the following file:

- source/umbra/lib/[UmbraMethodException.java](#)

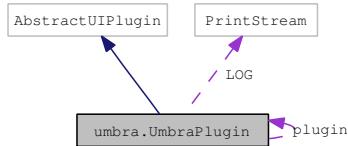
## 6.89 umbra.UmbraPlugin Class Reference

The main plugin class to be used in the desktop.

Inheritance diagram for umbra.UmbraPlugin:



Collaboration diagram for umbra.UmbraPlugin:



### Public Member Functions

- [UmbraPlugin \(\)](#)  
*The constructor which shares the instance.*
- final void [start](#) (final BundleContext a\_context) throws Exception  
*This method is called upon plug-in activation.*
- final void [stop](#) (final BundleContext a\_context) throws Exception  
*This method is called when the plug-in is stopped.*

### Static Public Member Functions

- static [UmbraPlugin getDefault \(\)](#)  
*Returns the shared instance.*
- static ImageDescriptor [getImageDescriptor](#) (final String a\_path)  
*Returns an image descriptor for the image file at the given plug-in relative path.*
- static void [messagelog](#) (final String a\_string)  
*This method prints out the string to the current logging facility.*

### Static Public Attributes

- static final PrintStream [LOG](#) = System.out  
*The standard logging facility for the plugin.*

## Static Private Attributes

- static [UmbraPlugin plugin](#)  
*The shared instance of the plugin.*

### 6.89.1 Detailed Description

The main plugin class to be used in the desktop.

#### Author:

Tomasz Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))  
Wojciech Was ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

#### Version:

a-01

### 6.89.2 Constructor & Destructor Documentation

#### 6.89.2.1 [umbra.UmbraPlugin.UmbraPlugin \(\)](#)

The constructor which shares the instance.

References [umbra.UmbraPlugin.plugin](#).

### 6.89.3 Member Function Documentation

#### 6.89.3.1 [final void umbra.UmbraPlugin.start \(final BundleContext \*a\\_context\*\) throws Exception](#)

This method is called upon plug-in activation.

##### Parameters:

*a\_context* the context from which the bundle for this plug-in is extracted

##### Exceptions:

*Exception* if this method fails to shut down this plug-in

#### 6.89.3.2 [final void umbra.UmbraPlugin.stop \(final BundleContext \*a\\_context\*\) throws Exception](#)

This method is called when the plug-in is stopped.

##### Parameters:

*a\_context* the context from which the bundle for this plug-in is extracted

##### Exceptions:

*Exception* if this method fails to shut down this plug-in

References [umbra.UmbraPlugin.plugin](#).

**6.89.3.3 static UmbraPlugin umbra.UmbraPlugin.getDefault () [static]**

Returns the shared instance.

**Returns:**

the only instance of the Umbra plugin in the system.

References umbra.UmbraPlugin.plugin.

**6.89.3.4 static ImageDescriptor umbra.UmbraPlugin.getImageDescriptor (final String *a\_path*) [static]**

Returns an image descriptor for the image file at the given plug-in relative path.

**Parameters:**

*a\_path* the path for the image

**Returns:**

the image descriptor

**6.89.3.5 static void umbra.UmbraPlugin.messageLog (final String *a\_string*) [static]**

This method prints out the string to the current logging facility.

**Parameters:**

*a\_string* the string to print to the log

References umbra.UmbraPlugin.LOG.

## 6.89.4 Member Data Documentation

**6.89.4.1 final PrintStream umbra.UmbraPlugin.LOG = System.out [static]**

The standard logging facility for the plugin.

Referenced by umbra.UmbraPlugin.messageLog().

**6.89.4.2 UmbraPlugin umbra.UmbraPlugin.plugin [static, private]**

The shared instance of the plugin.

Referenced by umbra.UmbraPlugin.getDefault(), umbra.UmbraPlugin.stop(), and umbra.UmbraPlugin.UmbraPlugin().

The documentation for this class was generated from the following file:

- source/umbra/[UmbraPlugin.java](#)

## 6.90 umbra.lib.UmbraRangeException Class Reference

This is an exception used to trace situations when the parsing exceeded some size associated with an exception.

### Public Member Functions

- [UmbraRangeException \(final Exception an\\_exception\)](#)  
*Creates an exception with the exception that caused the current one.*
- [Exception getCause \(\)](#)  
*Returns the exception which caused the current one.*

### Private Attributes

- [final Exception my\\_exception](#)  
*This field contains the exception which triggered the current one.*

### Static Private Attributes

- [static final long serialVersionUID = -5832574381679620026L](#)  
*The serial number of the class.*

#### 6.90.1 Detailed Description

This is an exception used to trace situations when the parsing exceeded some size associated with an exception.

It is used to encapsulate [UmbraMethodException](#) and [UmbraLocationException](#).

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

#### 6.90.2 Constructor & Destructor Documentation

##### 6.90.2.1 umbra.lib.UmbraRangeException.UmbraRangeException (final Exception *an\_exception*)

Creates an exception with the exception that caused the current one.

#### Parameters:

*an\_exception* the exception which caused the current one

References [umbra.lib.UmbraRangeException.my\\_exception](#).

### 6.90.3 Member Function Documentation

#### 6.90.3.1 Exception umbra.lib.UmbraRangeException.getCause()

Returns the exception which caused the current one.

**Returns:**

the exception which caused the current one

References umbra.lib.UmbraRangeException.my\_exception.

Referenced by umbra.lib.GUIMessages.exceededRangeInfo().

### 6.90.4 Member Data Documentation

#### 6.90.4.1 final long umbra.lib.UmbraRangeException serialVersionUID = -5832574381679620026L [static, private]

The serial number of the class.

#### 6.90.4.2 final Exception umbra.lib.UmbraRangeException.my\_exception [private]

This field contains the exception which triggered the current one.

Referenced by umbra.lib.UmbraRangeException.getCause(), and umbra.lib.UmbraRangeException.UmbraRangeException().

The documentation for this class was generated from the following file:

- source/umbra/lib/[UmbraRangeException.java](#)

## 6.91 umbra.lib.UmbraRuntimeException Class Reference

This is an exception used in reporting runtime exceptional events within Umbra.

### Public Member Functions

- [UmbraRuntimeException](#) (final String a\_string)

*Constructs a new Umbra runtime exception with the specified detail message.*

### Static Private Attributes

- static final long [serialVersionUID](#) = 4428245399391845887L

*The serial ID for the exception.*

### 6.91.1 Detailed Description

This is an exception used in reporting runtime exceptional events within Umbra.

This exception should not be handled either inside or outside Umbra.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.91.2 Constructor & Destructor Documentation

#### 6.91.2.1 umbra.lib.UmbraRuntimeException.UmbraRuntimeException (final String a\_string)

Constructs a new Umbra runtime exception with the specified detail message.

#### Parameters:

*a\_string* the message of the exception

### 6.91.3 Member Data Documentation

#### 6.91.3.1 final long umbra.lib.UmbraRuntimeException.serialVersionUID = 4428245399391845887L [static, private]

The serial ID for the exception.

The documentation for this class was generated from the following file:

- source/umbra/lib/[UmbraRuntimeException.java](#)

## 6.92 umbra.lib.UmbraSynchronisationException Class Reference

This is an exception used to trace situations when the synchronisation is attempted for a line in the source code document not in the code areas.

### Static Private Attributes

- static final long `serialVersionUID = 2772289259228267210L`

*The serial ID for the exception.*

### 6.92.1 Detailed Description

This is an exception used to trace situations when the synchronisation is attempted for a line in the source code document not in the code areas.

#### Author:

Aleksy Schubert ([alex@mimuw.edu.pl](mailto:alex@mimuw.edu.pl))

#### Version:

a-01

### 6.92.2 Member Data Documentation

#### 6.92.2.1 `final long umbra.lib.UmbraSynchronisationException serialVersionUID = 2772289259228267210L [static, private]`

The serial ID for the exception.

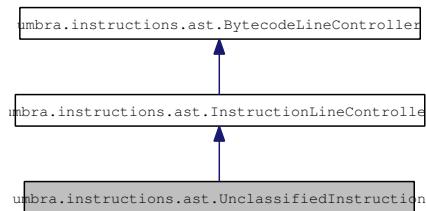
The documentation for this class was generated from the following file:

- source/umbra/lib/[UmbraSynchronisationException.java](#)

## 6.93 umbra.instructions.ast.UnclassifiedInstruction Class Reference

This is a class responsible for all lines which are regarded to be an instruction but unable to classify.

Inheritance diagram for umbra.instructions.ast.UnclassifiedInstruction:



Collaboration diagram for umbra.instructions.ast.UnclassifiedInstruction:



### Public Member Functions

- **UnclassifiedInstruction** (final String a\_line\_text, final String a\_name)
 

*This constructor creates an instruction which is not recognized.*
- final boolean **correct** ()
 

*Instruction out of classification is never correct.*
- boolean **needsMg** ()
 

*Returns true when a BCEL method representation must be associated with the current line controller.*

### Static Public Member Functions

- static String[ ] **getMnemonics** ()
 

*This method returns the array of unclassified instructions mnemonics.*

#### 6.93.1 Detailed Description

This is a class responsible for all lines which are regarded to be an instruction but unable to classify.

##### Author:

Tomek Batkiewicz ([tb209231@students.mimuw.edu.pl](mailto:tb209231@students.mimuw.edu.pl))  
 Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

##### Version:

a-01

### 6.93.2 Constructor & Destructor Documentation

#### 6.93.2.1 **umbra.instructions.ast.UnclassifiedInstruction.UnclassifiedInstruction (final String a\_line\_text, final String a\_name)**

This constructor creates an instruction which is not recognized.

It stores only the content of the line and the form of the mnemonic by calling the superclass constructor.

**Parameters:**

*a\_line\_text* the line with the unclassified mnemonic  
*a\_name* the unclassified mnemonic

**See also:**

[InstructionLineControllerInstructionLineController\(String, String\)](#)

### 6.93.3 Member Function Documentation

#### 6.93.3.1 **static String [] umbra.instructions.ast.UnclassifiedInstruction.getMnemonics () [static]**

This method returns the array of unclassified [instructions](#) mnemonics.

**Returns:**

the array of the handled mnemonics

**See also:**

[InstructionLineController.getMnemonics\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

#### 6.93.3.2 **final boolean umbra.instructions.ast.UnclassifiedInstruction.correct ()**

Instruction out of classification is never correct.

**Returns:**

`false`

**See also:**

[InstructionLineController.correct\(\)](#)

Reimplemented from [umbra.instructions.ast.InstructionLineController](#).

#### 6.93.3.3 **boolean umbra.instructions.ast.UnclassifiedInstruction.needsMg ()**

Returns `true` when a BCEL method representation must be associated with the current line controller.

In case of [UnclassifiedInstruction](#), this method returns always `false` as we do not know how to interpret these [instructions](#). Note that this means that [hasMg\(\)](#) results always in `false` as the method structure will never be assigned.

**Returns:**

true when a BCEL method representation must be associated with the current line controller, otherwise false

Reimplemented from [umbra.instructions.ast.BytecodeLineController](#).

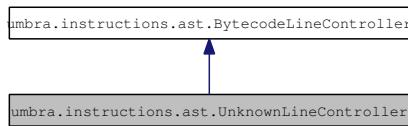
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[UnclassifiedInstruction.java](#)

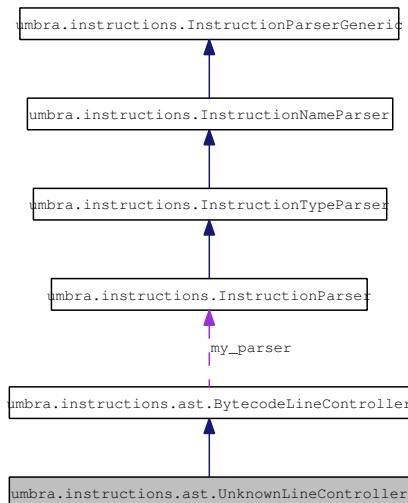
## 6.94 umbra.instructions.ast.UnknownLineController Class Reference

This class is responsible for all lines that we cannot classify.

Inheritance diagram for umbra.instructions.ast.UnknownLineController:



Collaboration diagram for umbra.instructions.ast.UnknownLineController:



### Public Member Functions

- [UnknownLineController](#) (final String a\_line\_text)

*This constructor only remembers the line with the unrecognized content.*

#### 6.94.1 Detailed Description

This class is responsible for all lines that we cannot classify.

##### Author:

Jarosław Paszek ([jp209217@students.mimuw.edu.pl](mailto:jp209217@students.mimuw.edu.pl))

##### Version:

a-01

## 6.94.2 Constructor & Destructor Documentation

### 6.94.2.1 umbra.instructions.ast.UnknownLineController.UnknownLineController (final String *a\_line\_text*)

This constructor only remembers the line with the unrecognized content.

#### Parameters:

*a\_line\_text* the string representation of the line with unrecognized content

#### See also:

BytecodeLineController.BytecodeLineController(String)

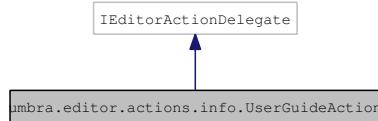
The documentation for this class was generated from the following file:

- source/umbra/instructions/ast/[UnknownLineController.java](#)

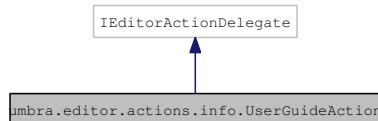
## 6.95 umbra.editor.actions.info.UserGuideAction Class Reference

The class implements the behaviour in case the User Guide button in the byte code `editor` is pressed.

Inheritance diagram for `umbra.editor.actions.info.UserGuideAction`:



Collaboration diagram for `umbra.editor.actions.info.UserGuideAction`:



### Public Member Functions

- void `setActiveEditor` (final `IAction` `an_action`, final `IEditorPart` `a_target_editor`)  
*The method sets the `editor` associated with the action.*
- final void `run` (final `IAction` `an_action`)  
*The method shows the content of the user guide `instructions`.*
- void `selectionChanged` (final `IAction` `an_action`, final `ISelection` `a_selection`)  
*The method reacts when the selected area changes in the byte code `editor`.*

### 6.95.1 Detailed Description

The class implements the behaviour in case the User Guide button in the byte code `editor` is pressed.

#### Author:

Wojciech Wąs ([ww209224@students.mimuw.edu.pl](mailto:ww209224@students.mimuw.edu.pl))

#### Version:

a-01

### 6.95.2 Member Function Documentation

#### 6.95.2.1 void `umbra.editor.actions.info.UserGuideAction.setActiveEditor` (final `IAction` `an_action`, final `IEditorPart` `a_target_editor`)

The method sets the `editor` associated with the action.

**Parameters:**

*an\_action* ignored

*a\_target\_editor* ignored

**6.95.2.2 final void umbra.editor.actions.info.UserGuideAction.run (final IAction *an\_action*)**

The method shows the content of the user guide [instructions](#).

Currently, it only pops up the general help browser.

FIXME the method should open something more specific, note that it is tricky to know the proper ID to open it should open something like Info/guide.txt <https://mobius.ucd.ie/ticket/556>

**Parameters:**

*an\_action* action that triggered the showing of the instruction

**6.95.2.3 void umbra.editor.actions.info.UserGuideAction.selectionChanged (final IAction *an\_action*, final ISelection *a\_selection*)**

The method reacts when the selected area changes in the byte code [editor](#).

Currently, it does nothing.

**Parameters:**

*an\_action* the action which triggered the selection change

*a\_selection* the new selection.

The documentation for this class was generated from the following file:

- source/umbra/editor/actions/info/[UserGuideAction.java](#)



# Chapter 7

## File Documentation

### 7.1 source/umbra/editor/actions/BytecodeColorAction.java File Reference

#### Namespaces

- namespace [umbra.editor.actions](#)

#### Classes

- class [umbra.editor.actions.BytecodeColorAction](#)

*This class defines an action of changing the coloring style.*

## 7.2 source/umbra/editor/actions/BytecodeCombineAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions](#)

### Classes

- class [umbra.editor.actions.BytecodeCombineAction](#)

*This is an action associated with a byte code [editor](#) (an extension .btc).*

## 7.3 source/umbra/editor/actions/BytecodeDoubleClickStrategy.java File Reference

### Namespaces

- namespace [umbra.editor.actions](#)

### Classes

- class [umbra.editor.actions.BytecodeDoubleClickStrategy](#)

*This class is responsible for action that is performed after a double click event in a byte code editor window.*

## 7.4 source/umbra/editor/actions/BytecodeEditorAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions](#)

### Classes

- class [umbra.editor.actions.BytecodeEditorAction](#)

*This class defines the common operations for all the byte code editor actions.*

## 7.5 source/umbra/editor/actions/BytecodeRebuildAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions](#)

### Classes

- class [umbra.editor.actions.BytecodeRebuildAction](#)

*This class defines action of restoring the original version of a class file (it is saved with the name prefixed with '\_') and then generating byte code (.btc) directly from it.*

## 7.6 source/umbra/editor/actions/BytecodeRefreshAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions](#)

### Classes

- class [umbra.editor.actions.BytecodeRefreshAction](#)

*This is a class defining an action: save current byte code editor window and re-generate byte code from the .class file.*

## 7.7 source/umbra/editor/actions/BytecodeSynchrAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions](#)

### Classes

- class [umbra.editor.actions.BytecodeSynchrAction](#)

*This class defines action of the synchronisation for a byte code position with an appropriate point in the source code.*

## 7.8 source/umbra/editor/actions/history/BytecodeRestoreAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions.history](#)

### Classes

- class [umbra.editor.actions.history.BytecodeRestoreAction](#)

*This class defines action of restoring byte code from [history](#).*

## 7.9 source/umbra/editor/actions/history/ClearHistoryAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions.history](#)

### Classes

- class [umbra.editor.actions.history.ClearHistoryAction](#)

*The bytecode [editor](#) action that removes all the historical versions of code.*

## 7.10 source/umbra/editor/actions/history/HistoryAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions.history](#)

### Classes

- class [umbra.editor.actions.history.HistoryAction](#)

*This class defines an action that adds current byte code snapshot to the [history](#) stack.*

## 7.11 source/umbra/editor/actions/info/InstalInfoAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions.info](#)

### Classes

- class [umbra.editor.actions.info.InstalInfoAction](#)

*The class implements the behaviour in case the Install Info button in the bytecode editor is pressed.*

## 7.12 source/umbra/editor/actions/info/UserGuideAction.java File Reference

### Namespaces

- namespace [umbra.editor.actions.info](#)

### Classes

- class [umbra.editor.actions.info.UserGuideAction](#)

*The class implements the behaviour in case the User Guide button in the byte code [editor](#) is pressed.*

## 7.13 source/umbra/editor/BytecodeConfiguration.java File Reference

### Namespaces

- namespace [umbra.editor](#)

### Classes

- class [umbra.editor.BytecodeConfiguration](#)

*This class is used by the [BytecodeEditor](#) with the matter of double click strategy and colour versions.*

## 7.14 source/umbra/editor/BytecodeContribution.java File Reference

### Namespaces

- namespace [umbra.editor](#)

### Classes

- class [umbra.editor.BytecodeContribution](#)

*This class represents a GUI element that is contributed to the eclipse GUI by the byte code editor.*

- class [umbra.editor.BytecodeContribution.BytecodeListener](#)

*This is a listener class that receives all the events that change the content of the current byte code document.*

## 7.15 source/umbra/editor/BytecodeDocument.java File Reference

### Namespaces

- namespace [umbra.editor](#)

### Classes

- class [umbra.editor.BytecodeDocument](#)

*This class is an abstract model of a byte code textual document.*

## 7.16 source/umbra/editor/BytecodeDocumentProvider.java File Reference

### Namespaces

- namespace [umbra.editor](#)

### Classes

- class [umbra.editor.BytecodeDocumentProvider](#)

*This class has been modified with relation to the generated automatically to allow adding listener that is responsible for error checking.*

## 7.17 source/umbra/editor/BytecodeEditor.java File Reference

### Namespaces

- namespace [umbra.editor](#)

### Classes

- class [umbra.editor.BytecodeEditor](#)

*This is the main class that defines the byte code [editor](#).*

## 7.18 source/umbra/editor/BytecodeEditorContributor.java File Reference

### Namespaces

- namespace [umbra.editor](#)

### Classes

- class [umbra.editor.BytecodeEditorContributor](#)

*This is managing class that adds [actions](#) to workbench menus and toolbars for a byte code [editor](#).*

## 7.19 source/umbra/editor/ColorModeContainer.java File Reference

### Namespaces

- namespace [umbra.editor](#)

### Classes

- class [umbra.editor.ColorModeContainer](#)

*This class is a static container that keeps the value of current colouring style that is obtained after each refreshing (which takes place when a byte code document is created too).*

## 7.20 source/umbra/editor/DocumentSynchroniser.java File Reference

### Namespaces

- namespace [umbra.editor](#)

### Classes

- class [umbra.editor.DocumentSynchroniser](#)

*This class handles the logic of the synchronisation of the cursor positions between the source code and the byte code documents.*

## 7.21 source/umbra/editor/parsing/BytecodeBMLSecScanner.java File Reference

### Namespaces

- namespace [umbra.editor.parsing](#)

### Classes

- class [umbra.editor.parsing.BytecodeBMLSecScanner](#)

*This class is responsible for colouring these texts in a byte code [editor](#) window which are inside BML annotations areas.*

## 7.22 source/umbra/editor/parsing/BytecodePartitionScanner.java

### File Reference

#### Namespaces

- namespace [umbra.editor.parsing](#)

#### Classes

- class [umbra.editor.parsing.BytecodePartitionScanner](#)

*This class is responsible for dividing the byte code document into partitions the colouring of which is governed by different rules.*

## 7.23 source/umbra/editor/parsing/BytecodeScanner.java File Reference

### Namespaces

- namespace [umbra.editor.parsing](#)

### Classes

- class [umbra.editor.parsing.BytecodeScanner](#)

*This class is responsible for colouring these texts in a byte code [editor](#) window which are outside BML annotations areas.*

## 7.24 source/umbra/editor/parsing/BytecodeWhitespaceDetector.java

### File Reference

#### Namespaces

- namespace [umbra.editor.parsing](#)

#### Classes

- class [umbra.editor.parsing.BytecodeWhitespaceDetector](#)

*This class defines objects that are able to check if a particular character is a whitespace.*

## 7.25 source/umbra/editor/parsing/BytecodeWordDetector.java File Reference

### Namespaces

- namespace [umbra.editor.parsing](#)

### Classes

- class [umbra.editor.parsing.BytecodeWordDetector](#)

*The class implements the way the words are scanned in the Eclipse scanners used in the Umbra [editor](#).*

## 7.26 source/umbra/editor/parsing/ColorManager.java File Reference

### Namespaces

- namespace [umbra.editor.parsing](#)

### Classes

- class [umbra.editor.parsing.ColorManager](#)

*This object manages the allocation and deallocation of the system colors that are used in the colouring in the bytecode editors.*

## 7.27 source/umbra/editor/parsing/ColorValues.java File Reference

### Namespaces

- namespace [umbra.editor.parsing](#)

### Classes

- class [umbra.editor.parsing.ColorValues](#)

*The interface defining colours used in particular colouring styles.*

## 7.28 source/umbra/editor/parsing/NonRuleBasedDamagerRepairer.java

### File Reference

#### Namespaces

- namespace [umbra.editor.parsing](#)

#### Classes

- class [umbra.editor.parsing.NonRuleBasedDamagerRepairer](#)

*This class is responsible for colouring these areas in a byte code editor window which are inside one-line areas.*

## 7.29 source/umbra/editor/parsing/SpecialNumberRule.java File Reference

### Namespaces

- namespace [umbra.editor.parsing](#)

### Classes

- class [umbra.editor.parsing.SpecialNumberRule](#)

*The text styling rule which extends the [NumberRule](#) so that it allows an additional mark before (and optionally after) the number to be read (used with '#' and '').*

## 7.30 source/umbra/editor/parsing/TokenGetter.java File Reference

### Namespaces

- namespace [umbra.editor.parsing](#)

### Classes

- class [umbra.editor.parsing.TokenGetter](#)

*This is an intermediary class which creates the Eclipse [parsing](#) and text partitioning classes with the properties established using the Umbra colouring modes.*

## 7.31 source/umbra/instructions/ast/AnnotationLineController.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.AnnotationLineController](#)

*This class handles the creation and correctness of line controllers that contain BML annotations.*

## 7.32 source/umbra/instructions/ast/ArithmeticInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.ArithmeticInstruction](#)

*This class handles the creation and correctness for arithmetic [instructions](#) with no parameters.*

## 7.33 source/umbra/instructions/ast/ArrayInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.ArrayInstruction](#)

*This class handles the creation and correctness for the instruction to create new arrays of primitive types (newarray).*

## 7.34 source/umbra/instructions/ast/BytecodeLineController.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.BytecodeLineController](#)

*This is completely abstract class that contains some information useful when the line is modified or BCEL structure is created.*

## 7.35 source/umbra/instructions/ast/CommentLineController.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.CommentLineController](#)

*This class handles the creation and correctness of line controllers that form comments.*

## 7.36 source/umbra/instructions/ast/ConversionInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.ConversionInstruction](#)

*This class handles the creation and correctness for the [instructions](#) with no parameters which convert types.*

## 7.37 source/umbra/instructions/ast/EmptyLineController.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.EmptyLineController](#)

*This is a class for a line with whitespaces only.*

## 7.38 source/umbra/instructions/ast/FieldInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.FieldInstruction](#)

*This class handles the creation and correctness for [instructions](#) to store and load field values.*

## 7.39 source/umbra/instructions/ast/HeaderLineController.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.HeaderLineController](#)

*This is a class for lines in bytecode files that occur at the beginning of methods.*

## 7.40 source/umbra/instructions/ast/IConstInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.IConstInstruction](#)

*This class handles the creation and correctness for iconst instructions with no parameters.*

## 7.41 source/umbra/instructions/ast/IincInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.IincInstruction](#)

*This class handles the creation and correctness for iinc instruction.*

## 7.42 source/umbra/instructions/ast/InstructionLineController.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.astInstructionLineController](#)

*This class defines a structure that describes a single byte code instruction and contains related BC<sub>E</sub>L structures.*

## 7.43 source/umbra/instructions/ast/InvokeInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.InvokeInstruction](#)

*This class handles the creation and correctness for invoke instructions.*

## 7.44 source/umbra/instructions/ast/JumpInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.JumpInstruction](#)

*This class handles the creation and correctness for jump instructions.*

## 7.45 source/umbra/instructions/ast/LdcInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.LdcInstruction](#)

*This class is related to some subset of [instructions](#) depending on parameters.*

## 7.46 source/umbra/instructions/ast/LoadStoreArrayInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.LoadStoreArrayInstruction](#)

*This class handles the creation and correctness for certain [instructions](#) with no parameters.*

## 7.47 source/umbra/instructions/ast/LoadStoreConstInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.LoadStoreConstInstruction](#)

*This class handles the creation and correctness for [instructions](#) with no parameters that perform loading and storing data on/from the operand stack.*

## 7.48 source/umbra/instructions/ast/MultiInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.MultiInstruction](#)

*This is abstract class for all [instructions](#) with at least one parameter.*

## 7.49 source/umbra/instructions/ast/NewInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.NewInstruction](#)

*This class handles the creation and correctness for [instructions](#) to create objects, check their types, and cast them, namely:.*

## 7.50 source/umbra/instructions/ast/NumInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.NumInstruction](#)

*This is abstract class for all [instructions](#) with a number as a parameter.*

## 7.51 source/umbra/instructions/ast/OtherInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.OtherInstruction](#)

*This is abstract class for all [instructions](#) which are correct with number parameter as well as with a string one (in "").*

## 7.52 source/umbra/instructions/ast/PushInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.PushInstruction](#)

*This class handles the creation and correctness for push instructions i.e.*

## 7.53 source/umbra/instructions/ast/SingleInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.SingleInstruction](#)

*This class handles the creation and correctness for certain [instructions](#) with no parameters.*

## 7.54 source/umbra/instructions/ast/StackInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.StackInstruction](#)

*This class handles the creation and correctness for load and store [instructions](#) with parameters i.e.*

## 7.55 source/umbra/instructions/ast/StringInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.StringInstruction](#)

*This is abstract class for all [instructions](#) with a string as a parameter.*

## 7.56 source/umbra/instructions/ast/ThrowsLineController.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.ThrowsLineController](#)

*This is a class for a special bytecode lines related to thrown exceptions, not to be edited by a user.*

## 7.57 source/umbra/instructions/ast/UnclassifiedInstruction.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.UnclassifiedInstruction](#)

*This is a class responsible for all lines which are regarded to be an instruction but unable to classify.*

## 7.58 source/umbra/instructions/ast/UnknownLineController.java File Reference

### Namespaces

- namespace [umbra.instructions.ast](#)

### Classes

- class [umbra.instructions.ast.UnknownLineController](#)

*This class is responsible for all lines that we cannot classify.*

## 7.59 source/umbra/instructions/BytecodeCommentParser.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.BytecodeCommentParser](#)

*This class handles the operations which are connected with the handling of the comments.*

## 7.60 source/umbra/instructions/BytecodeController.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.BytecodeController](#)

*This class defines some structures related to BCEL as well as to the byte code [editor](#) contents.*

## 7.61 source/umbra/instructions/BytecodeControllerComments.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.BytecodeControllerComments](#)

*This class contains the functionality of the [BytecodeController](#) class which is responsible for the handling of the end-of-line comments and interline comments.*

## 7.62 source/umbra/instructions/BytecodeControllerContainer.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.BytecodeControllerContainer](#)

*This class encapsulates the internal structures of the [BytecodeController](#) and gives the internal interface to them.*

## 7.63 source/umbra/instructions/BytecodeControllerHelper.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.BytecodeControllerHelper](#)

*This class contains various helper methods that are used in the [BytecodeController](#) class.*

## 7.64 source/umbra/instructions/BytecodeTextParser.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.BytecodeTextParser](#)

*This class handles the operations which are common to all the document parsers that are used in Umbra.*

## 7.65 source/umbra/instructions/CannotCallRuleException.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.CannotCallRuleException](#)

*This class is used to mark possible errors in quick dispatcher automaton [DispatchingAutomaton](#).*

## 7.66 source/umbra/instructions/DispatchingAutomaton.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.DispatchingAutomaton](#)

*This class implements an automaton which is used to quickly determine the type of the currently analysed line of the byte code text.*

## 7.67 source/umbra/instructions/FragmentParser.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.FragmentParser](#)

*This class is used to parse fragments of byte code textual documents.*

## 7.68 source/umbra/instructions/InitParser.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.InitParser](#)

*This class handles the initial parsing of a byte code textual document.*

## 7.69 source/umbra/instructions/InstructionNameParser.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.InstructionNameParser](#)

*This class is the part of the byte code instruction parser which contributes the parsing of various identifiers i.e.*

## 7.70 source/umbra/instructions/InstructionParser.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions InstructionParser](#)

*This class allows to parse the line with instruction.*

## 7.71 source/umbra/instructions/InstructionParserGeneric.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions InstructionParserGeneric](#)

*This class is the initial part of the byte code instruction parser class.*

## 7.72 source/umbra/instructions/InstructionParserHelper.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions InstructionParserHelper](#)

*This class contains helper methods that allow check the classes of various syntactical classes occurring in Java byte code files.*

## 7.73 source/umbra/instructions/InstructionTypeParser.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions InstructionTypeParser](#)

*This class is the part of the byte code instruction parser which contributes the parsing of various type representations.*

## 7.74 source/umbra/instructions/LineContext.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.LineContext](#)

*The line parser on which the parsing of the byte code textual representation is based is context sensitive.*

## 7.75 source/umbra/instructions/Preparsing.java File Reference

### Namespaces

- namespace [umbra.instructions](#)

### Classes

- class [umbra.instructions.Preparsing](#)

*This class handles the preparing of document lines.*

## 7.76 source/umbra/java/actions/CommitAction.java File Reference

### Namespaces

- namespace [umbra.java.actions](#)

### Classes

- class [umbra.java.actions.CommitAction](#)

*The action is used to commit changes made to Java source code.*

## 7.77 source/umbra/java/actions/DisasBCEL.java File Reference

### Namespaces

- namespace [umbra.java.actions](#)

### Classes

- class [umbra.java.actions.DisasBCEL](#)

*This class defines the action related to Java source [editor](#).*

## 7.78 source/umbra/java/actions/SynchrSBAction.java File Reference

### Namespaces

- namespace [umbra.java.actions](#)

### Classes

- class [umbra.java.actions.SynchrSBAction](#)

*This class defines an action of synchronization positions form source code to bytecode.*

## 7.79 source/umbra/lib/BMLParsing.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.BMLParsing](#)

*This class is responsible for communication with BMLLib library (except code position synchronization, that calls only stateless, static methods from BMLLib).*

## 7.80 source/umbra/lib/BytecodeStrings.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.BytecodeStrings](#)

*The container for all the byte code and BML strings.*

## 7.81 source/umbra/lib/BytecodeStringsGeneric.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.BytecodeStringsGeneric](#)

*The class is a container for all byte code [instructions](#) and the sequences that indicate the starts and ends of comments or BML annotation areas.*

## 7.82 source/umbra/lib/BytecodeStringsMnemonics.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.BytecodeStringsMnemonics](#)

*The container for all the byte code mnemonic strings.*

## 7.83 source/umbra/lib/EclipseIdentifiers.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.EclipseIdentifiers](#)

*This is just a container for textual Eclipse identifiers of objects defined in Umbra.*

## 7.84 source/umbra/lib/FileNames.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.FileNames](#)

*This is just container for operations on the file names used in the Umbra plugin (i.e.*

## 7.85 source/umbra/lib/GUIMessages.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.GUIMessages](#)

*This is just container for texts of all the GUI messages.*

## 7.86 source/umbra/lib/HistoryOperations.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.HistoryOperations](#)

*This class implements the operations on history items.*

## 7.87 source/umbra/lib/UmbraClassException.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.UmbraClassException](#)

*This is an exception used to trace situations when a problem with class file is encountered.*

## 7.88 source/umbra/lib/UmbraException.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.UmbraException](#)

*This is an exception used in tracing internal exceptional flows within Umbra.*

## 7.89 source/umbra/lib/UmbraLocationException.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.UmbraLocationException](#)

*This is an exception used to trace situations when the parsing exceeded the content of a document.*

## 7.90 source/umbra/lib/UmbraMethodException.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.UmbraMethodException](#)

*This is an exception used to trace situations when the processing reached a method which does not exist in the document being processed.*

## 7.91 source/umbra/lib/UmbraRangeException.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.UmbraRangeException](#)

*This is an exception used to trace situations when the parsing exceeded some size associated with an exception.*

## 7.92 source/umbra/lib/UmbraRuntimeException.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.UmbraRuntimeException](#)

*This is an exception used in reporting runtime exceptional events within Umbra.*

## 7.93 source/umbra/lib/UmbraSynchronisationException.java File Reference

### Namespaces

- namespace [umbra.lib](#)

### Classes

- class [umbra.lib.UmbraSynchronisationException](#)

*This is an exception used to trace situations when the synchronisation is attempted for a line in the source code document not in the code areas.*

## 7.94 source/umbra/UmbraPlugin.java File Reference

### Namespaces

- namespace [umbra](#)

### Classes

- class [umbra.UmbraPlugin](#)

*The main plugin class to be used in the desktop.*